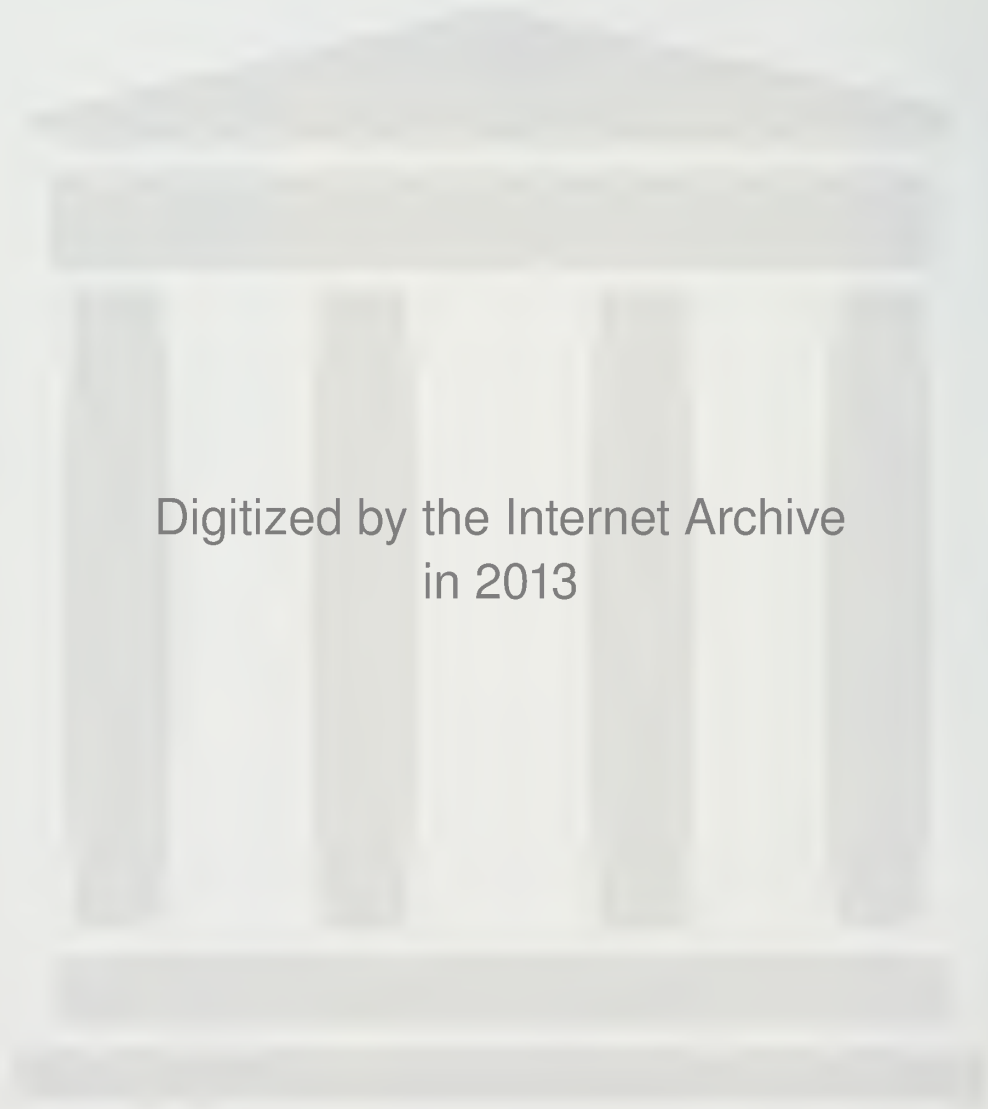


UIUCDCS-R-75-756

# A CLASS OF COMPACT HIGH SPEED PARALLEL MULTIPLICATION SCHEMES

by

WILLIAM JOHN STENZEL

September, 1975

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-75-756

# A CLASS OF COMPACT HIGH SPEED PARALLEL MULTIPLICATION SCHEMES

BY

WILLIAM JOHN STENZEL

September, 1975

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

This work was supported in part by Grant No. US NASA NAS5-23334 and was submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer science at the University of Illinois.

## ACKNOWLEDGMENT

The author would like to express his deepest gratitude to his thesis advisor, Professor W. J. Kubitz, for his interest, guidance, encouragement, and, above all, patience over the past several years.

The author would also like to thank Professor D. L. Slotnick for providing the invaluable opportunity to participate in this project, and who, through his constructive criticisms and questionable sense of humor, provided the stimulus for the development of the prototype multiplier.

Additionally, thanks are due to Marvin Graham and Gilles Garcia for their continuing advice and support and to Frank Serio and the Electronics Fabrication Group, without whom the construction of the prototype would have been impossible.

TABLE OF CONTENTS

# LIST OF FIGURES

# 1. INTRODUCTION

Various forms of parallel multiplication have been proposed. The schemes are roughly divisible into 2 classes--those which consist of an iterative array of cells [Advanced Micro Devices 25S05, Chung, Deegan, Fairchild 9344, Perzaris] and those which entail the generation of a matrix of partial product terms and the subsequent reduction of the matrix by means of pseudo adders [Dadda, Habibi, Singh, Svoboda, Wallace]. Array schemes are attractive in that they are fairly compact and often involve only one circuit type but their speed of operation increases linearly with the length of the operands to be multiplied and hence is slow for large words. Matrix generation-reduction schemes are much faster for large operands since their speed of operation increases with the log of the operand length. Traditional forms of the matrix generation-reduction scheme employ AND gates as 1 by 1 bit multipliers in forming the partial product matrix and use full adders to reduce the matrix; this form is not nearly as compact as the array scheme. Current LSI technology has made possible integrated circuits which can form partial products for operands larger than one bit [Texas Instruments 74S274] as well as ICs which can reduce larger portions of the matrices than is possible with full adders [Kingsbury, Texas Instruments 74S275] thus allowing fabrication of generation-reduction type multipliers which rival array type multipliers in compactness.

This paper attempts to deal with compact forms of generation-reduction type multipliers. The consequences of employing larger partial

product generation circuits and more general reduction circuits are considered; possible implementations of these circuits are discussed. An algorithm for the design of multipliers using these circuits is presented and is used to obtain gross measures of merit for several multiplication schemes. Finally, the fabrication of a prototype 24 x 24 bit multiplier employing such circuits will be described and its performance will be evaluated. Note that all schemes described are for unsigned (i.e., sign-magnitude) multiplication. If 2's complement multiplication is desired, the algorithm presented by Baugh and Wooley may be used to generate additional partial product terms which will produce a 2's complement result when either added to the unsigned product or incorporated into the matrix and reduced along with the rest of the terms [Baugh].

## 2. BASIC TECHNIQUES

Consider two n bit unsigned binary numbers, A and B, of the form $A = \sum_{i=0}^{n-1} a_i 2^i$, $B = \sum_{i=0}^{n-1} 6_i 2^i$. We may form the product of these numbers by first calculating the $n^2$ product terms $p_{ij} = a_i * b_j$ for i, j = 0, ..., n-1 with weighting factor $2^{i+j}$, and then summing the $n^2$ terms to form the 2n bit product. The form that this operation assumes is shown diagramatically in Figure 2.1. This is analogous to long-hand multiplication. The product terms, $p_{ij}$, may be readily computed by using AND gates as 1 x 1 bit multipliers--$p_{ij} = a_i * b_j = a_i b_j$. Hence the difficulty lies in forming the sum of the matrix. The approach followed by Wallace and Dadda used full adders to reduce three bits of weight $2^i$ to one bit of weight $2^i$ and one bit of weight $2^{i+1}$. This is commonly termed Wallace tree reduction. This reduction is shown diagramatically in Figure 2.2 as a dot matrix in which the three encircled input bits yield two output bits. The basis of the scheme is the repeated application of this reduction until a matrix with at most two terms present in a given column is obtained. This two row matrix is then reduced to a single row (i.e., the product) by means of a standard carry look-ahead adder. An example for the 6 x 6 case is shown in Figure 2.3. Note that the trapeziodal representation of the $n^2$ matrix has been altered to a triangular form. The matrix of height six has been reduced by full adders to height four, then by another set of adders to height three, and finally to height two. Carry look-ahead adders reduce this to the final product.

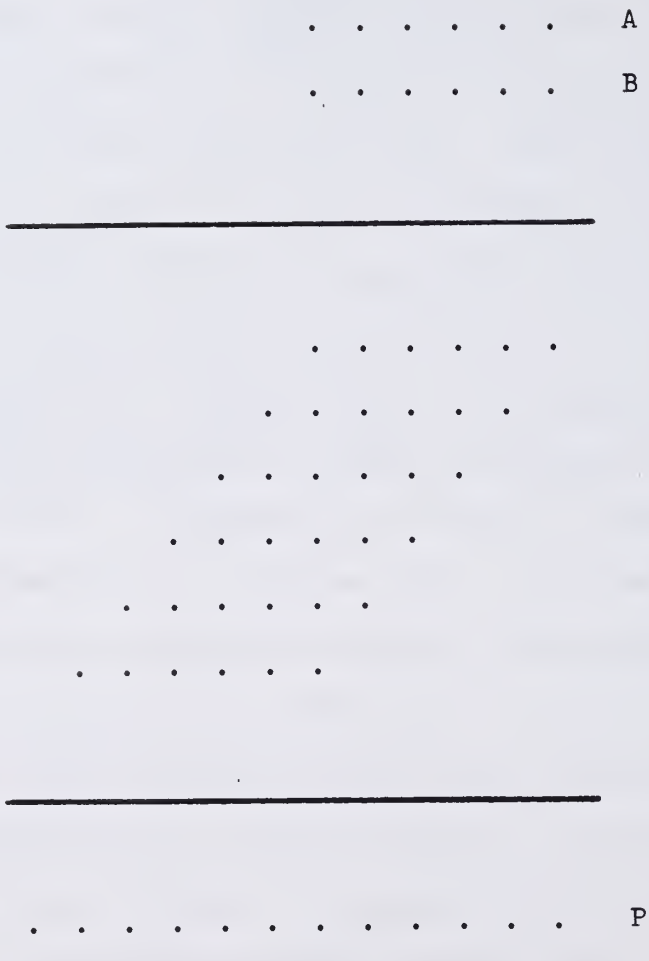



Figure 2.1 Diagramatic Representation of 6 x 6 Multiplication

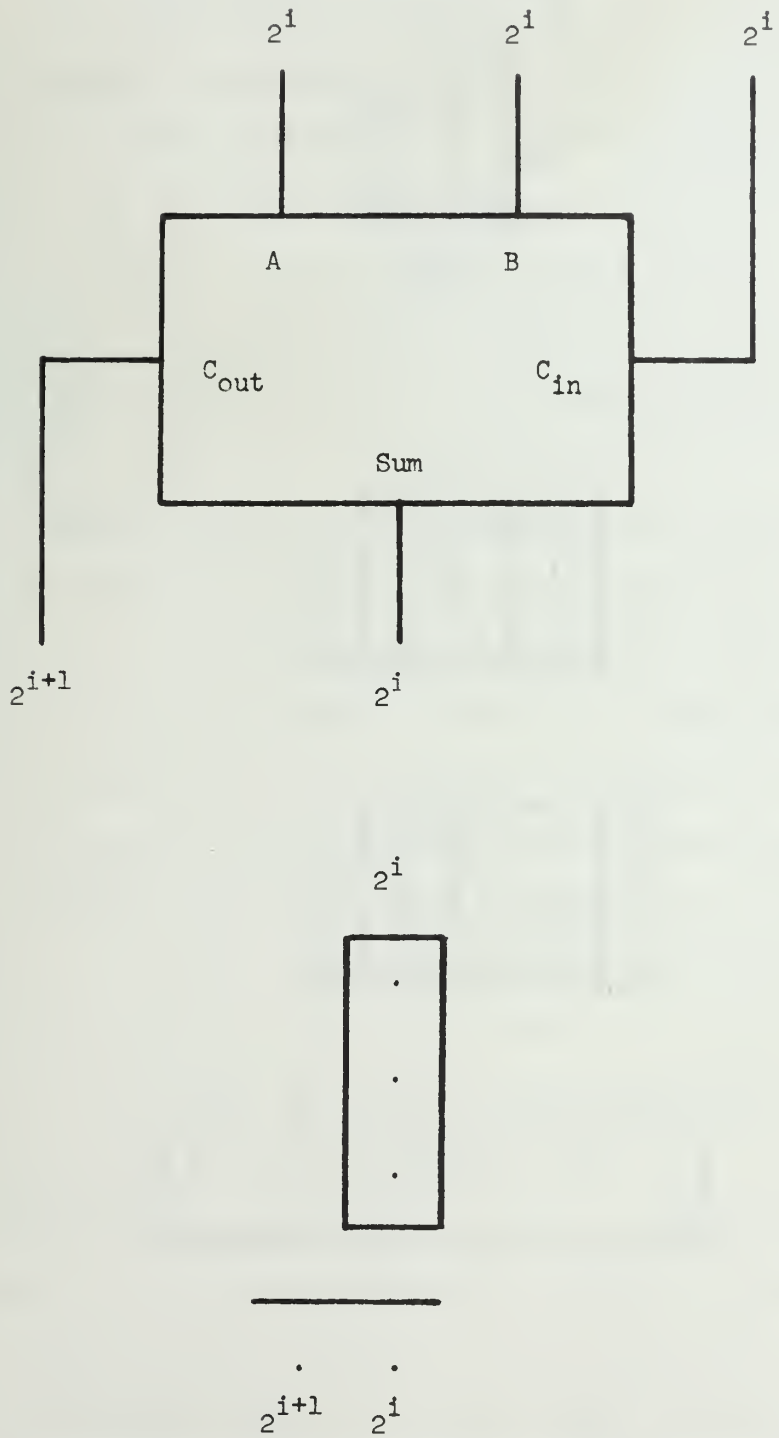



Figure 2.2 Full Adder as Used to Reduce Columns of Bits

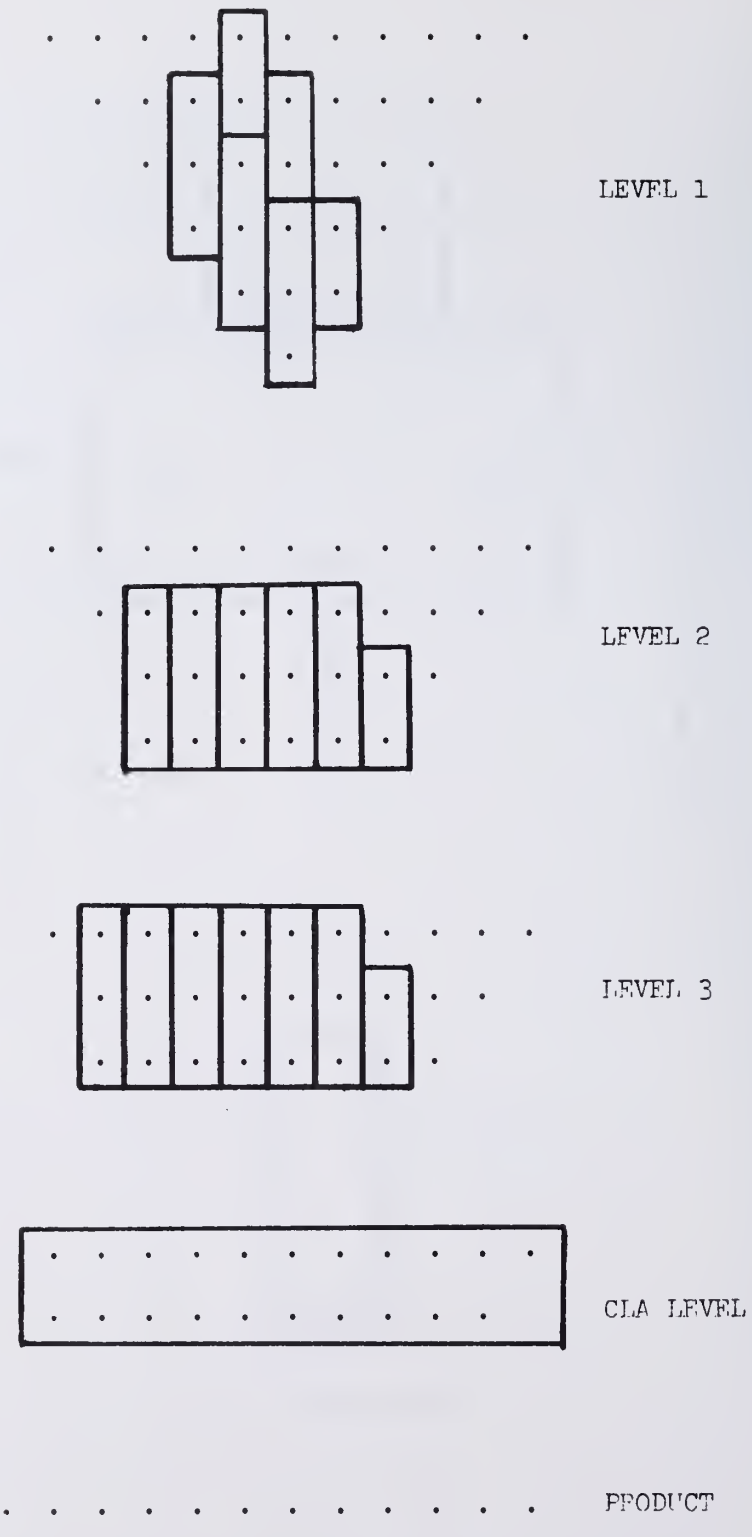



Figure 2.3 Wallace Tree Reduction for 6 x 6 Multiplication

This scheme may be improved by the use of a more sophisticated means of product term generation. There exist currently 256 x 8 read only memories (ROMs) (Figure 2.4) which may be programmed with the multiplication tables for pairs of 4 bit operands. That is, the two 4 bit operands are input as an 8 bit address at which location their 8 bit product is stored. Hence, the ROMs may be used as 4 x 4 multipliers in generating 8 bit partial product terms, rather than employing AND gates as 1 x 1 multipliers.

The applicability of this approach may be demonstrated by detailing the generation of the partial product matrix for a 12 x 12 bit multiplication. First consider the partial product generation for a 4 x 12 bit multiplication, as shown in Figure 2.5. Three terms of 8 bits each are generated ($A_0B_0$ thru $A_2B_0$), and are expressed more compactly in the form labeled $AB_0$. Now consider the partial product generation for the full 12 x 12 bit multiplication, as shown in Figure 2.6. It consists of three sets of terms similar to $AB_0$ which may in turn be compactly expressed in the triangular form shown which is 5 bits high at its deepest point and 24 bits wide. The full $n^2$ expansion would have been 12 bits high and contained exactly twice as many terms.

The savings thus obtained in partial product generation leads one to seek similar savings in the matrix reduction. Accordingly, we may note that currently available 1K x 4 bit ROMs may be programmed to treat the 10 address lines as 2 adjacent columns of 5 bits each and perform a table lookup on the sum of all the bits (Figure 2.7). A ROM so

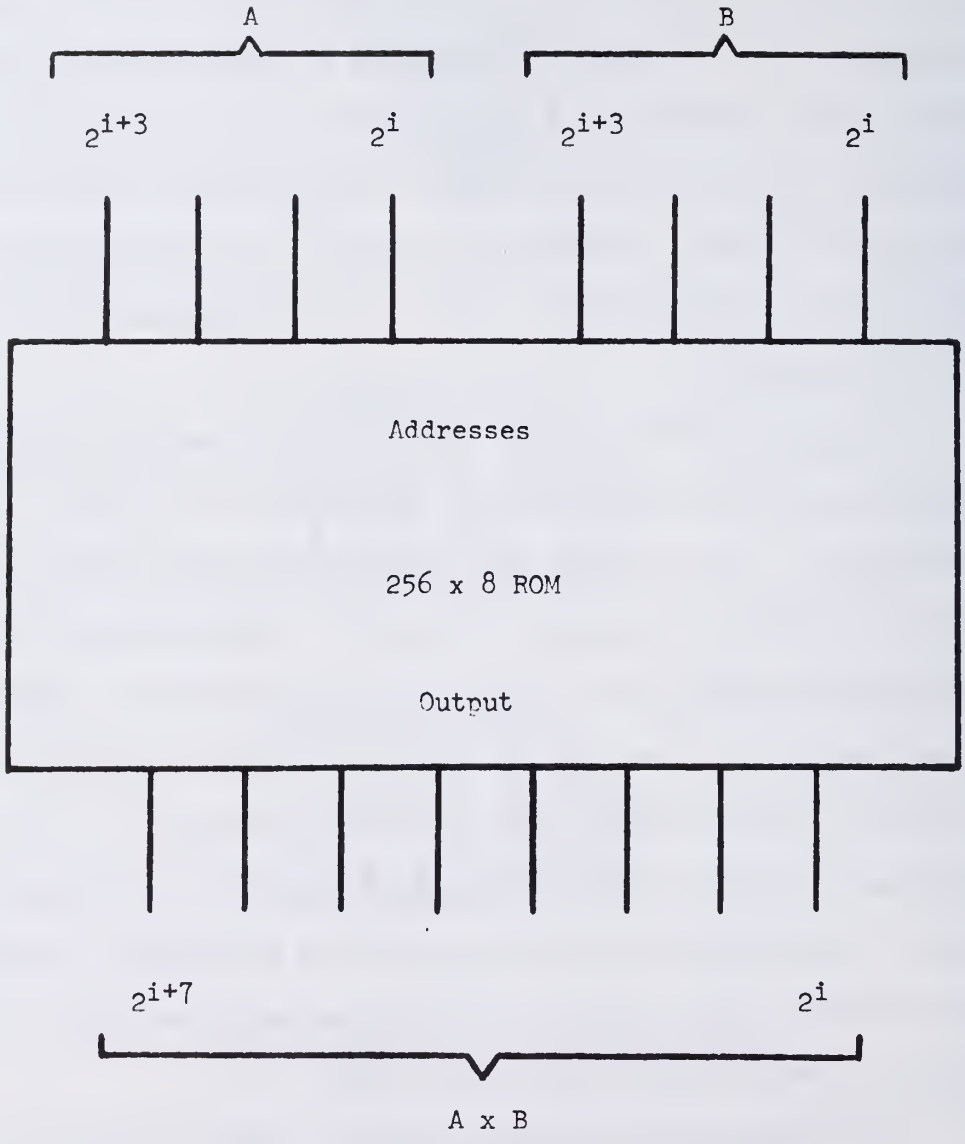



Figure 2.4 2k ROM as 4 x 4 Multiplier

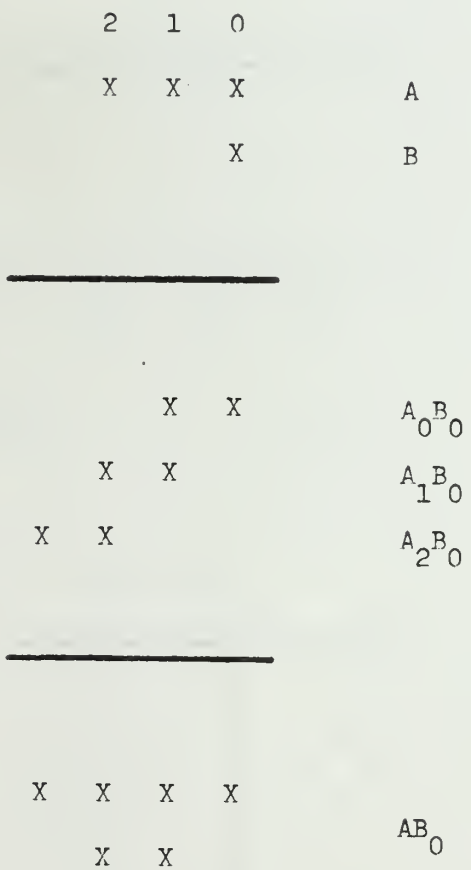



Figure 2.5 4 x 12 Bit Partial Product Matrix (Each 'X' represents four bits.)

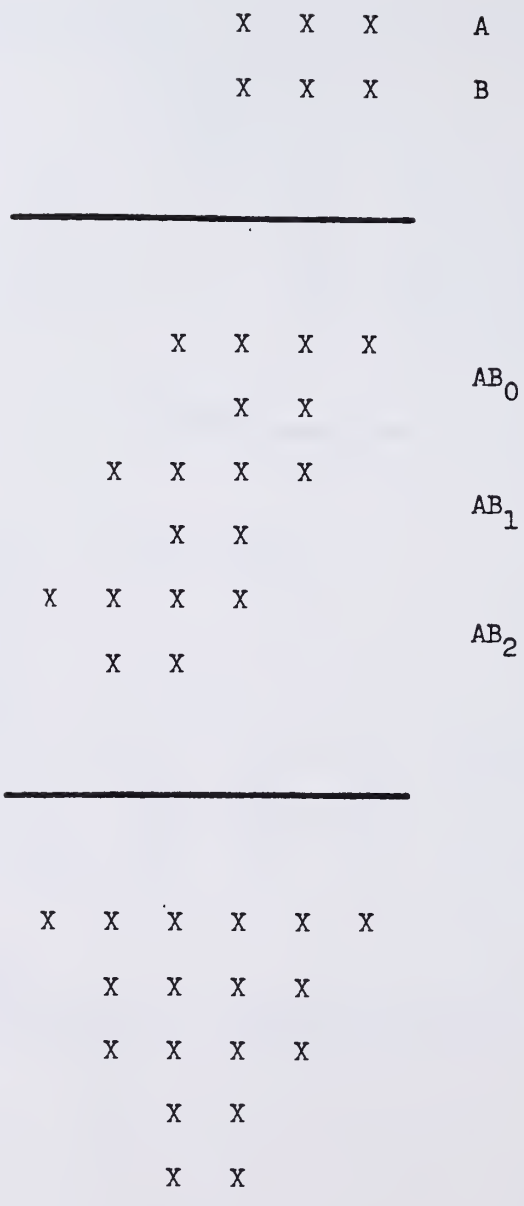



Figure 2.6 12 x 12 Bit Partial Product Matrix (Each 'X' represents four bits.)

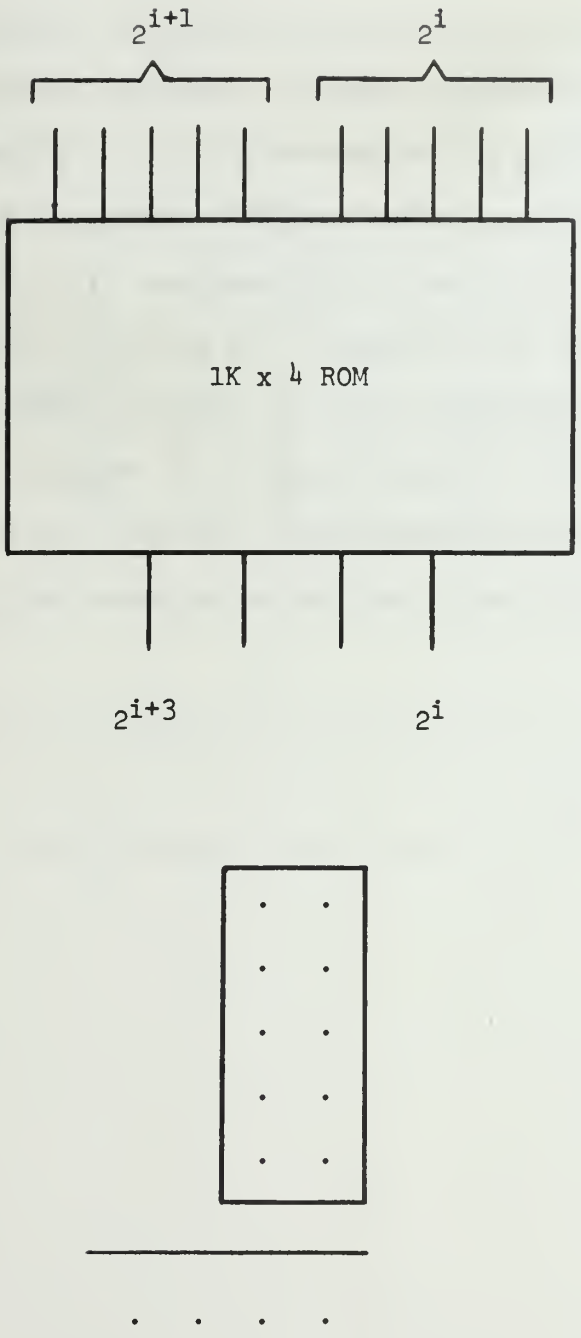



Figure 2.7 1k x 4 ROM as (5,5,4) Counter

programmed could be used to reduce a matrix in a fashion similar to the full adders of the previous example. Note that the maximum sum of the first column is five, and the maximum sum of the adjacent column is twice that, or ten; hence the maximum possible total sum is 15, which requires exactly 4 bits for its binary representation. This provides complete utilization of the 10 bit address field and of the 4 bit output field while providing a convenient tool for reducing large, symmetric portions of the matrix. This tool may be applied to the matrix of Figure 2.6 with the result shown in Figure 2.8. The first level in Figure 2.8 corresponds to the full matrix and the second shows the resulting two row matrix.

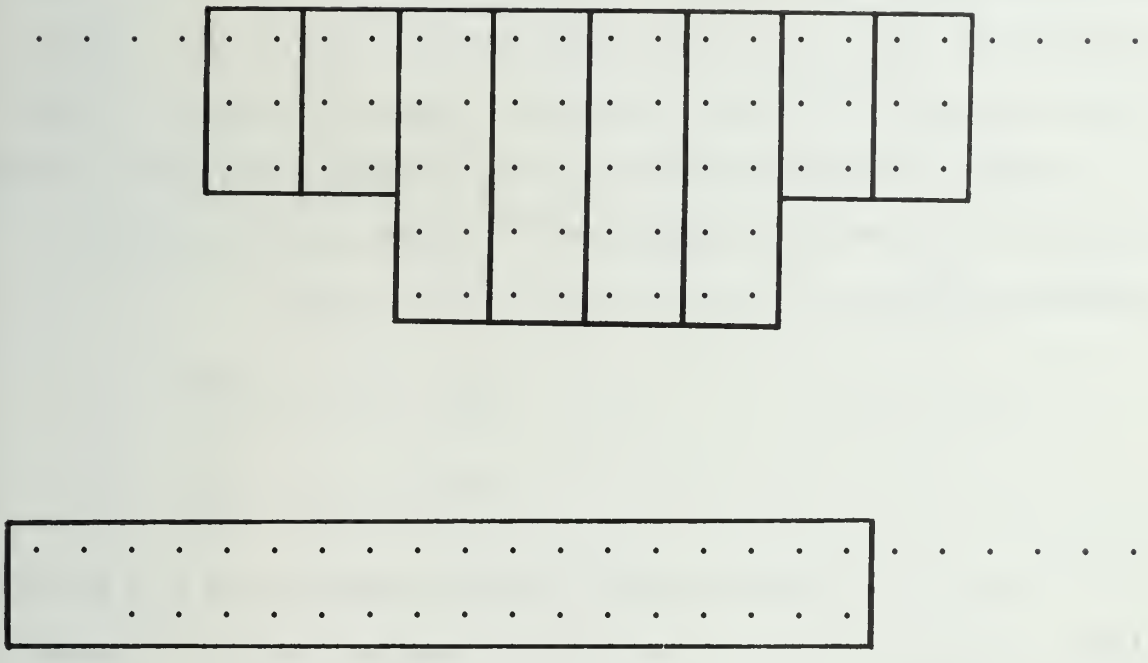


Figure 2.8 12 x 12 Bit Partial Product Reduction
Using (5,5,4) Counters

## 3. M x M MULTIPLIERS

Consider the generation of the partial product tree for an n by n bit multiplication using 1 x 1 bit multipliers. A total of $n^2$ partial products of one bit each are generated via $n^2$ multipliers. These partial products align themselves in the triangular form shown in Figure 3.1 in which each dot represents one bit. The depth at the center of the matrix is n bits and the height of the $i^{th}$ column (i.e., the column weighted by $2^i$) is

$$h_i = i + 1 \qquad \text{for } i = 0, \ldots, n-1$$

$$h_i = 2n - 1 - i \qquad \text{for } i = n, \ldots, 2n-2.$$

Now let us consider partial product generation via m by m multipliers with $m \geq 2$. We may divide the n bit operands into $n' = \frac{n}{m}$ segments of m bits (assuming for the sake of simplicity that n is divisible by m) and form the partial product tree by taking the $(n')^2$ crossproducts of the segments using $(n')^2$ multiplier modules. Each of the crossproducts contains 2m bits, hence the matrix contains a total of

$$2m(n')^2 = 2m(\frac{n}{m})^2 = \frac{2n^2}{m}$$

bits. If we let each dot in Figure 3.1 represent an m bit segment rather than a single bit, the figure shows the alignment of the low order halves of the partial products. Since the height of individual columns is constant within m bit blocks, we may denote the height of a block of m columns

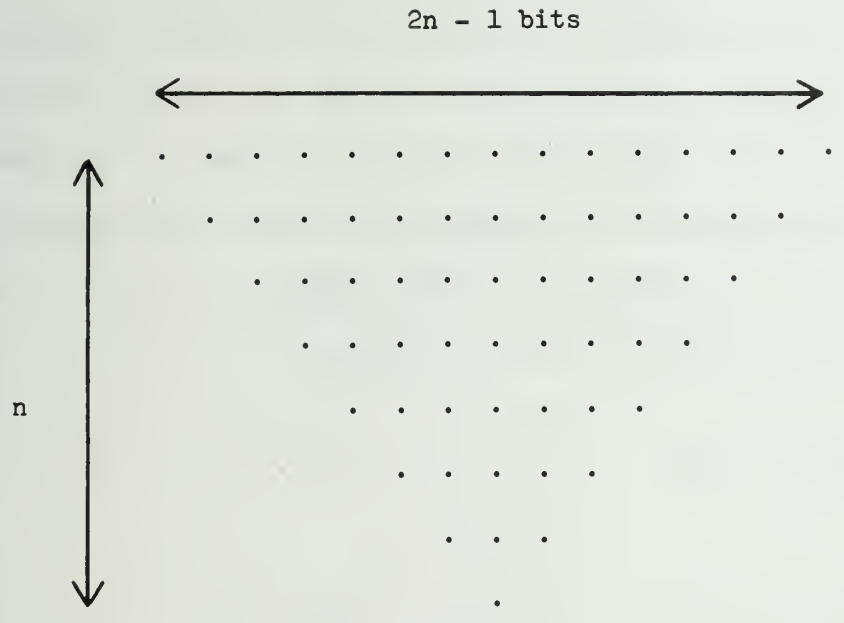



Figure 3.1 Partial Product Tree for 1 x 1 Multipliers

(for which the lowest order column has weighting $2^{mi}$) as $g_i$. If we consider only the contributions of the low order portions of the partial products, the height of the blocks of columns will be given by

$$g_i' = i + 1 \qquad \text{for } i = 0, \ldots, n'-1$$

$$g_i = 2n' - 1 - i \qquad \text{for } i = n', \ldots, 2n' -2.$$

Now let us take into account the effect of the high order portion of the products. Note that if the low order portion of a product contributes to block i-1 its high order will contribute to block i. This means that the total height of block i is equal to the number of low order segments in block i-1 plus the number of low order segments in block i. Denoting the total height of block i as $g_i'$ this gives

$$g_i' = g_i + g_{i-1} \qquad \text{for } i = 0, \ldots, 2n' -1$$

or

$$g_i' = (i + 1) + i = 2i + 1 \qquad \text{for } i = 0, \ldots, n'-1$$

$$g_i' = (2n' - 1 - i) + (2n' - i)$$

$$= 4n' - 1 - 2i \qquad \text{for } i = n', \ldots, 2n'-1.$$

The matrix has the form shown in Figure 3.2, i.e., the height at the extreme high and low order blocks of the matrix is 1 and the height increases by 2 every m bits moving toward the center. The height at the center is

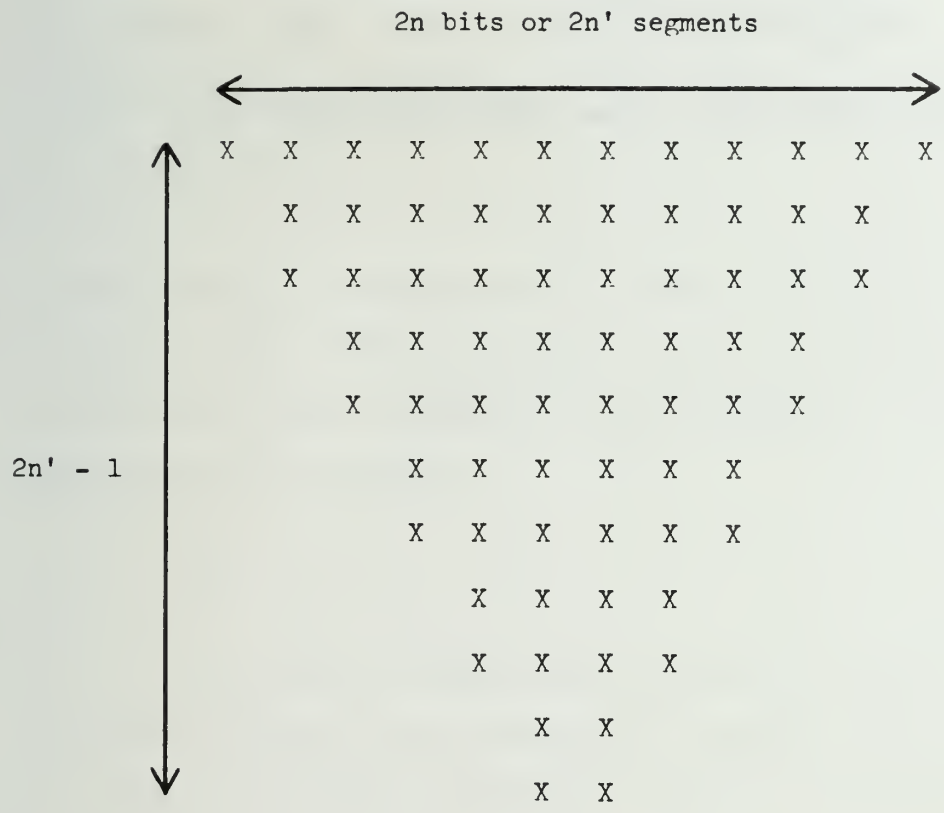



Figure 3.2 Partial Product Tree for m x m Multipliers
(Each 'X' represents m bits.)

$$2n' - 1 = \frac{2n}{m} - 1.$$

The height of the individual columns may be equivalently expressed as

$$h_i = 2\left\lfloor \frac{i}{m} \right\rfloor - 1 \qquad \text{for } i = 0, \ldots, n-1$$

$$h_i = 2\left\lfloor \frac{(2n - 1 - i)}{m} \right\rfloor + 1 \qquad \text{for } i = n, \ldots, 2n-1$$

where $h_i$ denotes the height of the column with weighting $2^i$.

## 4. GENERALIZED COUNTERS

Dadda has introduced the notion of a (c,d) counter as a combinatorial network which receives c bits of equal weight (i.e., a single column of c bits) as input and produces a d bit word corresponding to their sum as output. A full adder, for example, would be termed a (3,2) counter. The value of the output, then may be expressed as

$$v = \sum_{i=0}^{m-1} b_i$$

where $b_i$ denotes the binary value of bit i of the input column and v denotes the value of the d bit output. Note that the number of output bits must be sufficient to represent all possible sums of c bits, and hence must satisfy the condition

$$2^d - 1 \geq c.$$

This class of counters may readily be extended to include counters which receive several successively weighted input columns and produce their sum, taking the weighting into account. We may denote counters of this type as

$$(c_{k-1}, c_{k-2}, \ldots, c_0, d)$$

counters, where k is the number of input columns, $c_i$ is the number of input bits in the column of weight $2^i$, and d is the number of bits in the output word. The value of the output may be expressed as

$$v = \sum_{i=0}^{k-1} \sum_{j=0}^{c_i-1} b_{ij}\, 2^i$$

where $b_{ij}$ denotes the value of bit j in column i. The number of bits in the output word must again be sufficient to represent the largest possible sum, hence d is subject to the constraint that

$$2^d - 1 \geq \sum_{i=0}^{k-1} c_i\, 2^i.$$

Examples of several counters are shown in Figure 4.1 in dot matrix form. The encircled dots represent the configuration of the input bits, and are followed by the resultant output bits.

Consider now the effect of a series of counters acting on adjacent sets of input columns, as shown in Figure 4.2. The inputs to the counters are shown first, followed by the resulting output bits, followed in turn by an equivalent but more compact expression of the output bits. Let us refer to the number of resulting output rows as s. We see, then, that a series of (7,3) counters can reduce a matrix 7 rows high to a matrix 3 rows high (s = 3), or a series of (5,5,4) counters can reduce 5 rows to 2 rows (s = 2). Note also that a string of (2,2,2,3,5) counters can reduce 3 rows to 2 rows by virtue of the fact that an extra input bit is consumed in the low order position, compensating for the carry out of the previous counter.

Let us now focus our attention on counters with input columns of equal height. Such counters provide a convenient tool for reducing regular portions of a matrix and, in general, are more interesting and useful than irregular counters. The regularity of the counters permits

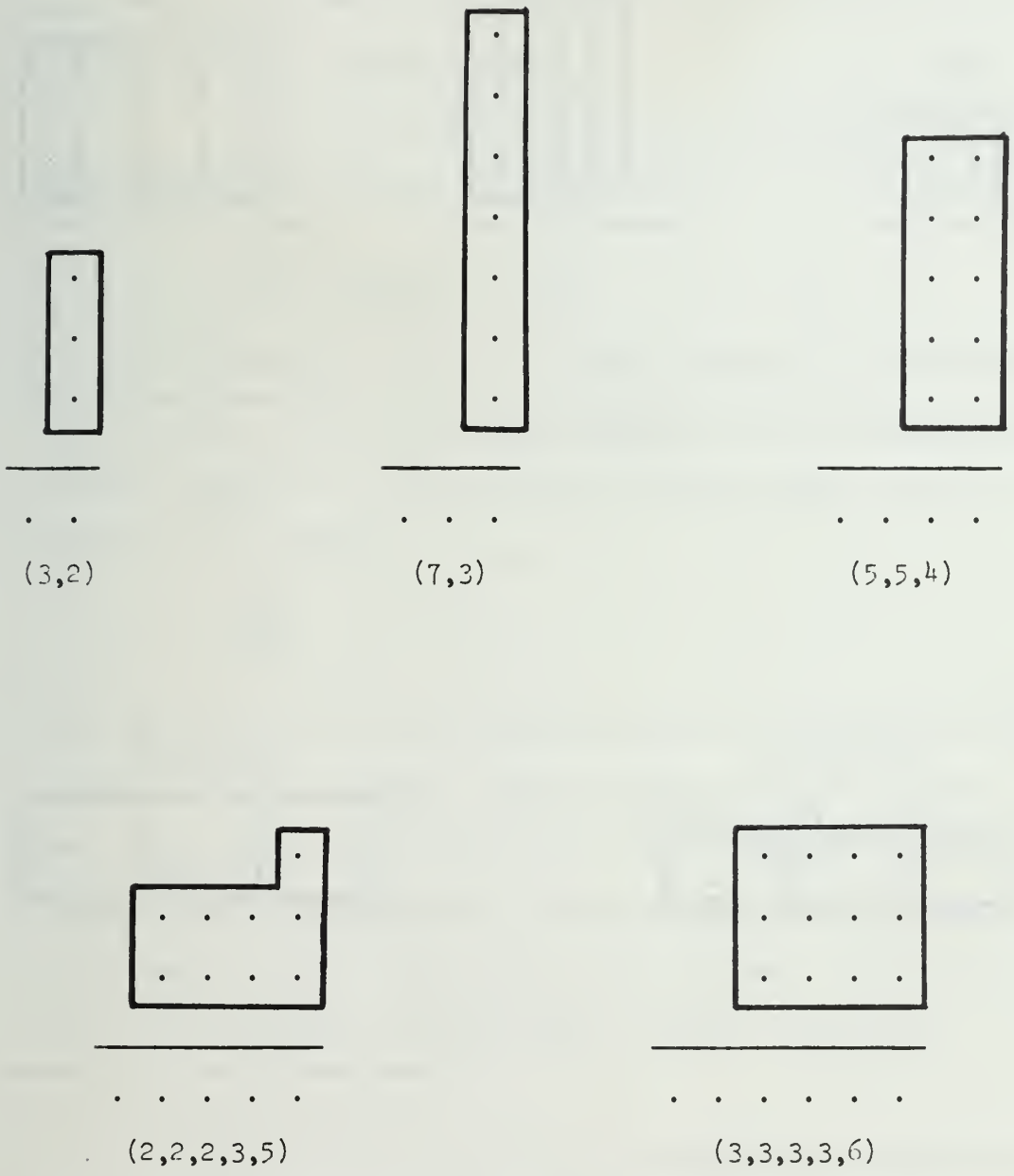



Figure 4.1 Some Generalized Counters

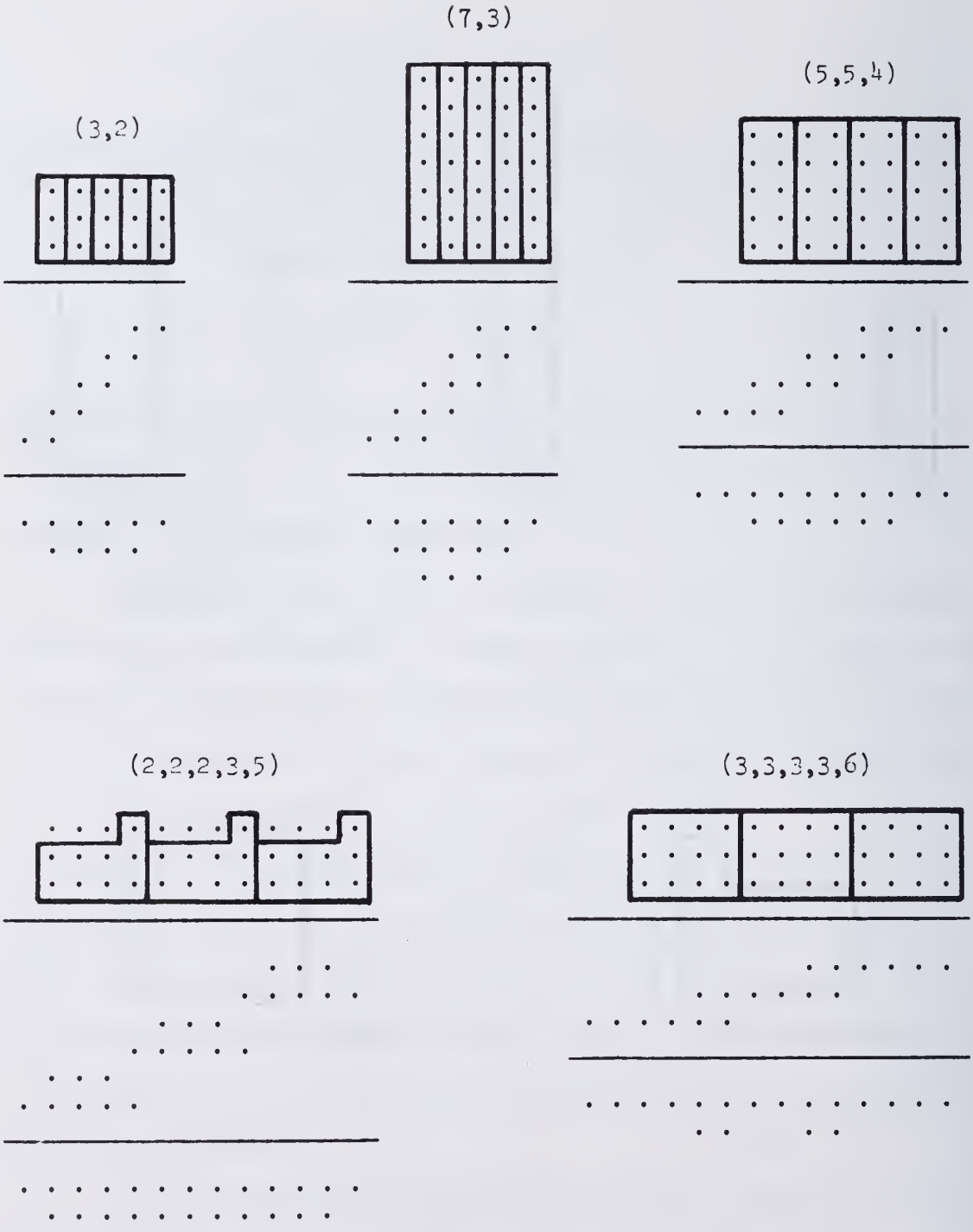



Figure 4.2 Effect of a Series of Adjacent Counters

us to make the following observations about them, which are not necessarily true of unequal column counters.

A counter with equal height columns will consume a rectangular matrix segment of k columns by r rows, where

$$r = c_{k-1} = c_{k-2} = \ldots = c_0.$$

As Figure 4.2 shows, a string of counters produces d bit outputs at intervals of k bits. The outputs align themselves such that no more than $\left\lceil \frac{d}{k} \right\rceil$ outputs contribute to a given column, hence the number of rows, s, of the output matrix is simply

$$s = \left\lceil \frac{d}{k} \right\rceil .$$

Thus the height of the resulting output matrix is determined by the number of input and output columns, subject to the constraint that the sum of the r input rows is representable in d bits. Note that the number of resultant output rows has a direct bearing on the number of stages of counters needed to reduce a large matrix (as explained in the next section), hence it is desirable that the number of output rows be small. If we let

$$v_r = 2^k - 1$$

denote the maximum possible value of a single input row and let

$$v_o = 2^d - 1$$

denote the maximum representable output value, the constraint on the number of input rows may be expressed as

$$v_o \geq r\, v_r$$

or

$$r \leq \frac{v_o}{v_r} = \frac{2^d - 1}{2^k - 1} .$$

Note that if d is not divisible by k, the output matrix will be somewhat sparse, e.g., the output of a string of (3,3,3,3,6) counters consists of alternate 2 bit segments of height 1 and height 2. This may be used to advantage in certain situations, such as the reduction of a six row matrix by means of (3,3,3,3,6) counters. The reduction is accomplished by stacking two strings of counters, one atop the other, as shown in Figure 4.3. Note that direct alignment of the counters produces a ragged matrix of height 4, but that skewing the 2 strings of counters by 2 bits produces a matrix which is uniformly 3 bits high. The advantage of producing an output matrix of uniform height is, of course, that it can in turn be easily reduced by a single counter type.

These considerations lead us to the notion of a maximally efficient counter type as a counter which produces a uniform output matrix while consuming the largest possible regular portion of a matrix, i.e., it should have equal columns with the largest possible number of rows. Disregarding cases where strings of counters may be stacked and skewed as in the above example, this means that the number of output columns should be a multiple of the number of input columns, or

$$d = s\, k.$$

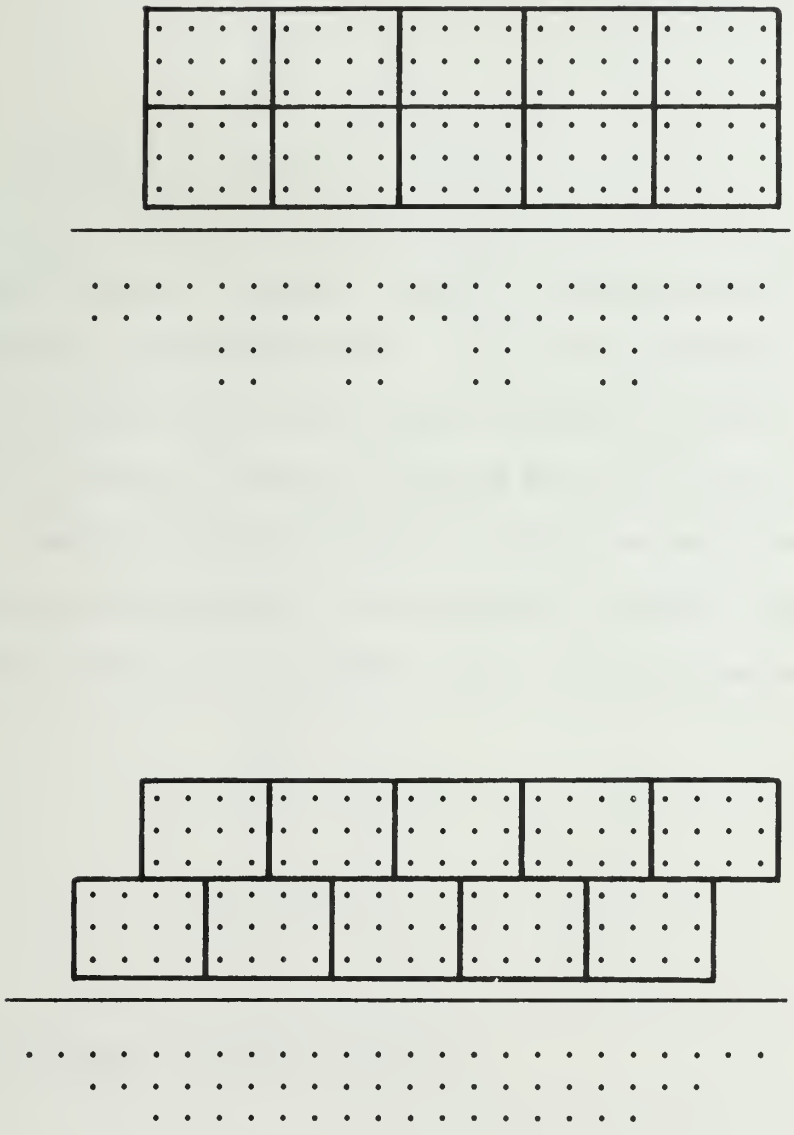


Figure 4.3 Six Row Matrix Reduction Using (3,3,3,3,6) Counters

The counter will consume the largest possible portion of the matrix when

$$v_o = r\, v_r$$

so that

$$r = \frac{v_o}{v_r} = \frac{2^d - 1}{2^k - 1} = \frac{2^{sk} - 1}{2^k - 1} = 2^{(s-1)k} + 2^{(s-2)k} + \ldots + 1.$$

A table of such maximally efficient counters is shown in Figure 4.4 for the first few values of k and s. The only currently implementable such maximally efficient counters having equal input columns are the (3,2), the (7,3), the (5,5,4) and possibly the (15,4) counters. The number of input rows increases rapidly with both the number of input columns and output rows, hence any implementation of counters with a larger k or s than those mentioned above would probably be less than maximally efficient.

| | INPUT | | OUTPUT | |
|---|---|---|---|---|
| | Columns | Rows | Columns | Rows |
| | k | r | d | s |
| * | 1 | 3 | 2 | 2 |
| * | 2 | 5 | 4 | 2 |
| | 3 | 9 | 6 | 2 |
| | 4 | 17 | 8 | 2 |
| | 5 | 33 | 10 | 2 |
| * | 1 | 7 | 3 | 3 |
| | 2 | 21 | 6 | 3 |
| | 3 | 73 | 9 | 3 |
| | 4 | 273 | 12 | 3 |
| * | 1 | 15 | 4 | 4 |
| | 2 | 85 | 8 | 4 |
| | 3 | 585 | 12 | 4 |
| | 1 | 31 | 5 | 5 |
| | 2 | 341 | 10 | 5 |

* denotes currently realizable counters

Figure 4.4 Maximally Efficient Counters with Equal Columns

## 5. NUMBER OF LEVELS FOR REDUCTION

In the previous section it was shown that a matrix r bits high could be reduced to one s bits high through the use of one level of counters. We may now turn to the issue of the number of levels of counters required to reduce a matrix more than r bits in height to one of s bits. Let us denote the maximum height of the matrix which may be reduced to s bits using j levels of counters as $l_j$. Note that $l_0 = s$ and $l_1 = r$. Knowing $l_j$ we may determine $l_{j+1}$ by observing that the $l_j$ bits represents the output of a stack of $\left\lfloor \frac{l_j}{s} \right\rfloor$ counters at level j+1 plus a residue of $(l_j)$ mod s bits which were not reduced by counters (Figure 5.1). The $\left\lfloor \frac{l_j}{s} \right\rfloor$ counters each consume r bits, so

$$l_{j+1} = r \left\lfloor \frac{l_j}{s} \right\rfloor + (l_j) \bmod s.$$

For a (5,5,4) counter, then, we have the maximum reduction height sequence

2, 5, 11, 26, 65, ...

and for a (7,3) counter we have the sequence

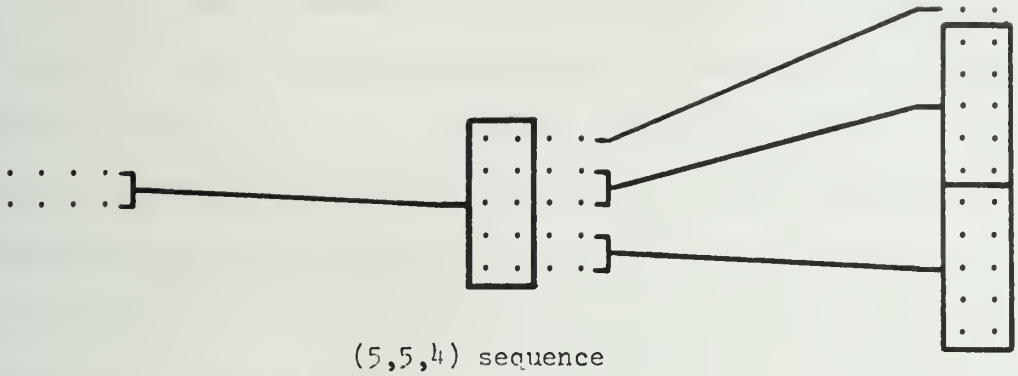
3, 7, 15, 35, 79, ...

A rough approximation of the first few terms of the sequence is

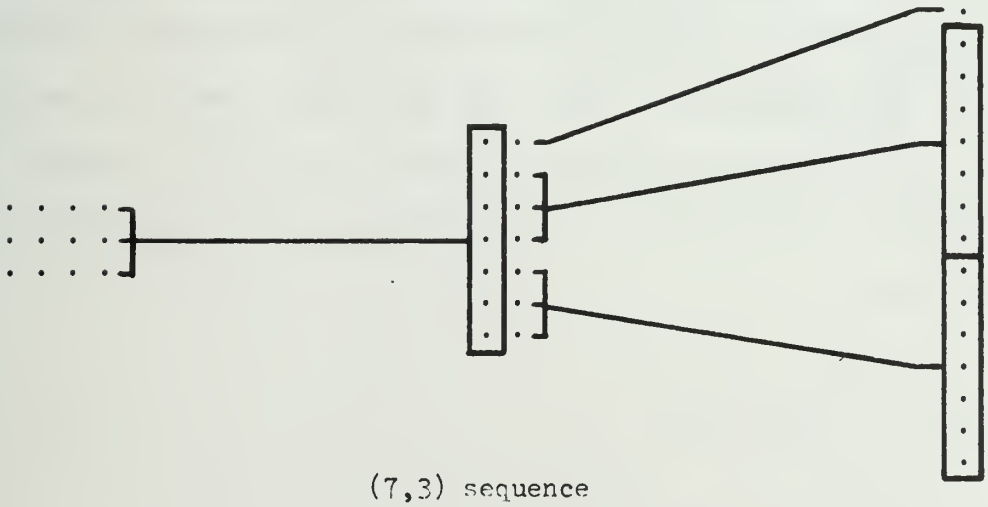
$$s,\ s(\tfrac{r}{s}),\ s(\tfrac{r}{s})^2,\ s(\tfrac{r}{s})^3,\ \ldots$$

hence the number of levels of counters required to reduce a matrix h bits high to s rows is roughly $\log_{(\frac{r}{s})}(h)$ levels.

Note that not all counters give a reduction to 2 rows--the (7,3) counter for example will give a reduction to only 3 rows. The final



(5,5,4) sequence



(7,3) sequence

Figure 5.1 Examples of Multilevel Reduction by (5,5,4) and (7,3) Counters

reduction to 2 rows must be accomplished by means of an additional stage of counters, e.g., (3,2) or (2,2,2,3,5) counters. Note also that other sequences may be obtained by combining different types of counters. For example, if the final level of a series of (5,5,4) counters is followed by an additional level of (3,2) counters, the final reduction height and hence the first term of the sequence for the (5,5,4) counters becomes 3 rather than 2. The sequence for the (3,2) level is 2, 3 and for the (5,5,4) levels is 3, 6, 15, 36, 90, ... giving an overall sequence of 2, 3, 6, 15, 36, .... Thus, if we wish to employ (5,5,4) counters for the reduction of a matrix which is initially 15 bits high to 2 rows we may use either 3 levels of (5,5,4) counters or 2 levels of (5,5,4) counters followed by one level of (3,2) counters. Assuming that (3,2) counters have a shorter propagation delay than (5,5,4) counters, the second approach will be slightly faster than the first. Similarly, we may insert a level of (7,3) counters between a final level of (3,2) counters and succeeding levels of (5,5,4) counters (Figure 5.2). The sequence for the (3,2) level is

2,3

for the (7,3) level its

3, 7

and for the (5,5,4) levels it is

7, 16, 40, 100, ...

Combining these, we get an overall sequence of

2, 3, 7, 16, 40, 100, ...

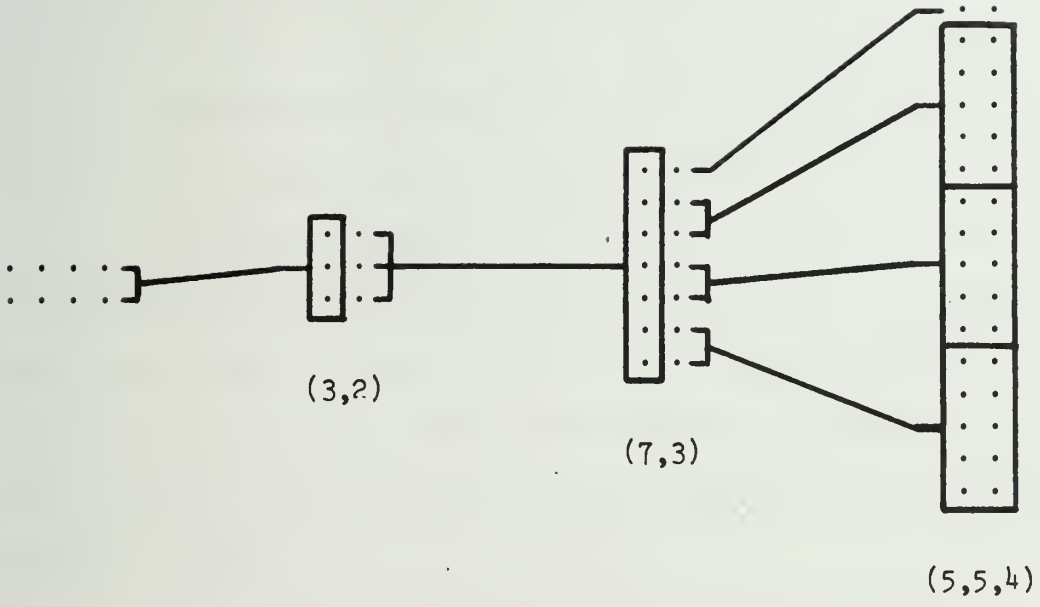



Figure 5.2 Effect of Mixing Counter Types Upon the Reduction Sequence

Hence we may in some sense tailor a sequence to fit the requirements of a particular matrix by mixing various counter types.

## 6. REDUCTION ALGORITHM

The heuristic rule for the reduction of a matrix with (3,2) counters as presented by Dadda is directly applicable to reduction by generalized counters. Dadda's rule was to find the largest term $l_j$ in the counter's sequence which is less than the original matrix height and to reduce the matrix only to height $l_j$. Each successive level of counters then reduces the matrix to the height given by the next smaller term in the series, i.e., $l_j$, $l_{j-1}$, ..., $l_0$. In the more general situation the sequence may be characteristic of more than one type of counter, as in the examples of the previous section.

The actual placement of the counters at a given reduction level is determined by traversing the matrix from right to left, inserting counters as needed to reduce columns to the targeted reduction height. The entire process may be stated algorithmically as follows (it is assumed that the choice of counters has been made, and that the reduction sequence is known):

> Denote the desired word length as n, the multiplier module size as m (i.e., an m by m multiplier), and the counters at a given level as ($c_{k-1}$, ...$c_0$, d) as before, where k is the number of input columns, d is the number of output columns, and $s = \left\lceil \frac{d}{k} \right\rceil$ is the number of output rows. Further let $h_i$ for i = 0, ..., 2n-1 denote the current height of column i of the matrix and let the variables H and T denote the matrix height before and

after the reduction, respectively; i.e., T is the targeted reduction height for a given level.

1. Initialize $h_i$ and H as follows:

   a) if m = 1

   $$h_i = i + 1 \qquad \text{for } i = 0, \ldots, n - 1$$

   $$h_i = 2n - 1 - i \qquad \text{for } i = n, \ldots, 2n - 1$$

   $$H = n$$

   b) if $m \geq 2$

   $$h_i = 2 \left\lfloor \frac{i}{m} \right\rfloor + 1 \qquad \text{for } i = 0, \ldots, n - 1$$

   $$h_i = 2 \left\lfloor \frac{(2n - 1 - i)}{m} \right\rfloor + 1 \qquad \text{for } i = n, \ldots, 2n - 1$$

   $$H = \frac{2n}{m} - 1$$

2. Set T to the largest term in the reduction series which is less than H, i.e.,

   $$T = l_j \qquad \text{for } j \text{ such that } l_j < H \text{ and } l_{j+1} \geq H$$

3. If H = 2 then terminate the algorithm; otherwise perform step 3a for i = 0, ..., 2n - 1.

   a) If $h_i \leq T$ do nothing; otherwise insert a counter at this point, adjust the column heights accordingly, and repeat step 3a for the new height of the same

column, i. The rules for adjusting column heights are:

$$h_{i+j} = \max [T, (h_{i+j} - c_j + 1)] \quad \text{for } j = 0, \ldots, k - 1$$

$$h_{i+j} = h_{i+j} + 1 \quad \text{for } j = k, \ldots, d - 1$$

4. Set H to T and T to the next smaller term in the reduction series. This term is given by

$$T = s \left\lfloor \frac{T}{r} \right\rfloor + (T) \bmod r$$

where the s and r values characterize the counters which will be used at the next level.

5. Go to step 3.

This process lends itself to solution via computer. Accordingly, subroutines have been written which design multipliers using various combinations of counters for word sizes of 4 to 64 bits. These routines are used to determine the numerical results presented later.

An example of a reduction is shown in Figure 6.1 in which 32 x 32 bit multiplication is performed using 4 x 4 bit multipliers for partial product generation and (5,5,4) counters for reduction. The height of the matrix is initially 15 and the counter's sequence is 2, 5, 11, 26. Hence the matrix is first reduced to height 11, then to height 5, and finally to height 2. Note that in some cases a (5,5,4) counter has been underutilized so that the reduction proceeds only to the desired height.

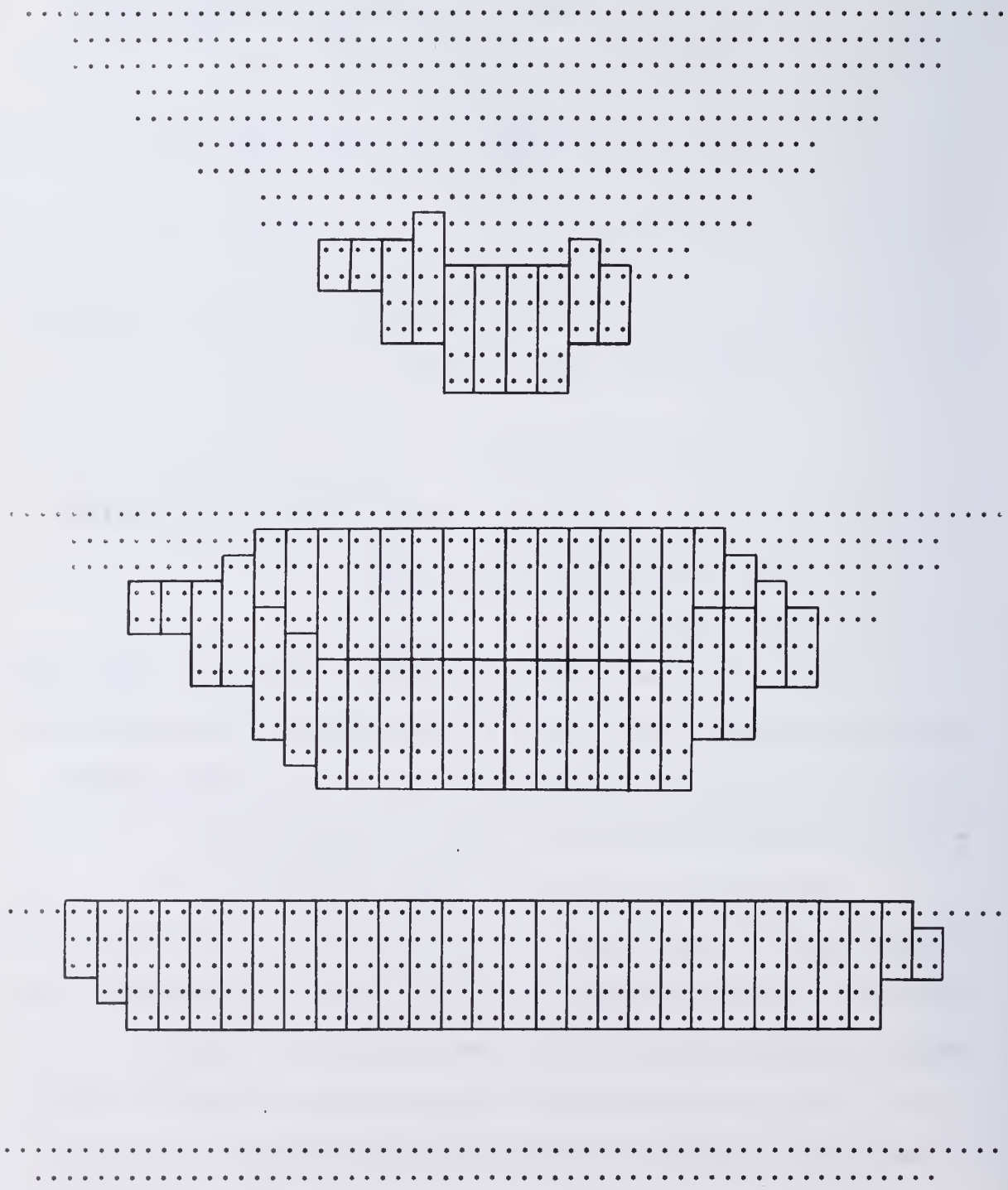


Figure 6.1 32 x 32 Bit Multiplication Using Only (5,5,4) Counters

In the previous section it was shown that a reduction from 15 rows to 2 rows could be accomplished by using 2 levels of (5,5,4) counters and one level of (3,2) counters. Hence an alternative reduction scheme is presented in Figure 6.2. We follow the same rules for reduction after altering the sequence to

2, 3, 6, 15.

The matrix is reduced first to height 6 and then to height 3 by (5,5,4) counters, and finally to height 2 by (3,2) counters. If there is a need to economize on the number of large counters (e.g., if the cost of a (5,5,4) counter is much larger than that of a (3,2) counter) a slight minimization can be obtained by using smaller counters to replace underutilized large counters wherever possible. The effect of such a minimization is shown in Figure 6.3.

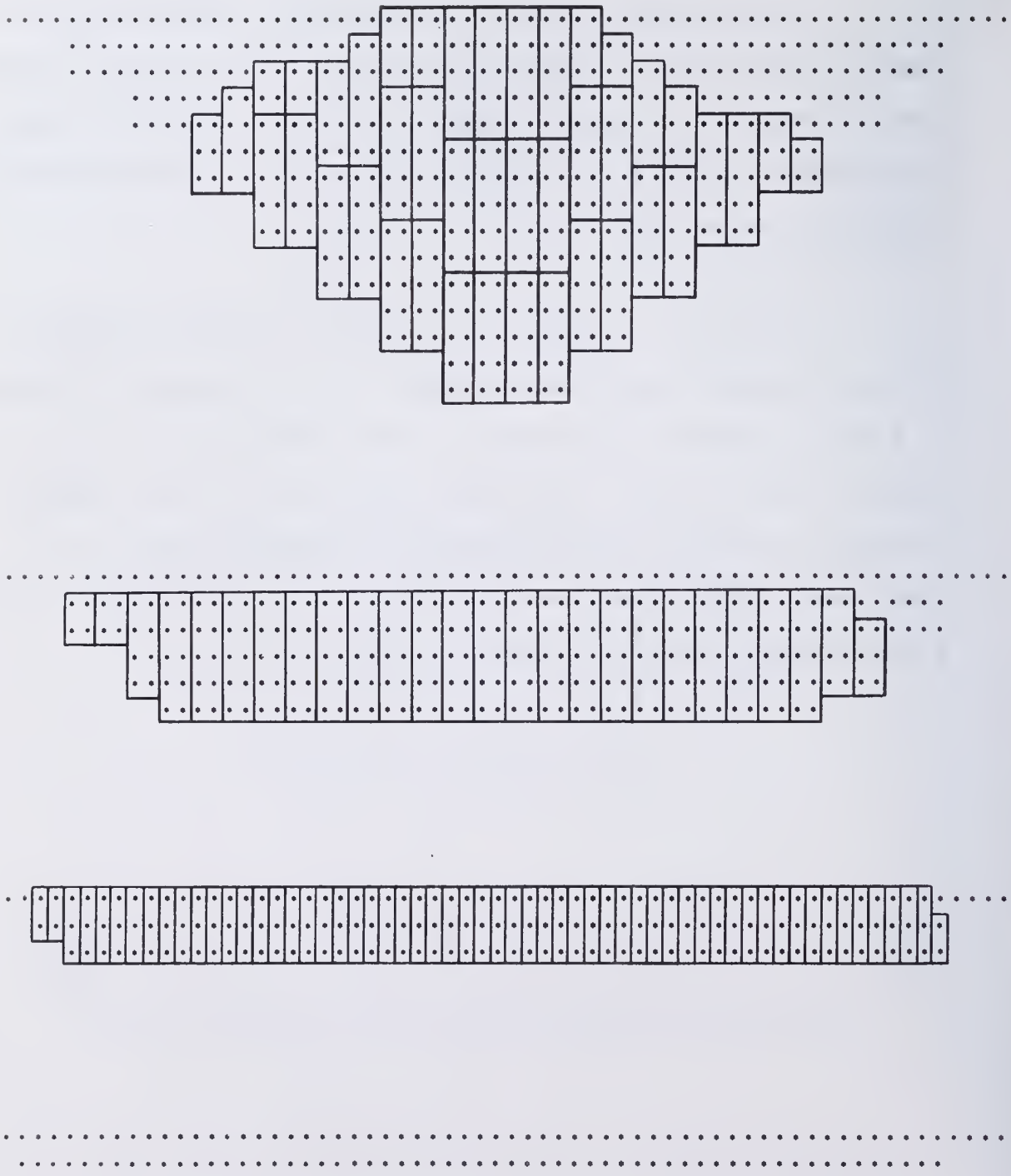


Figure 6.2 32 x 32 Bit Multiplication Using (5,5,4) with (3,2) Counters in Last Stage

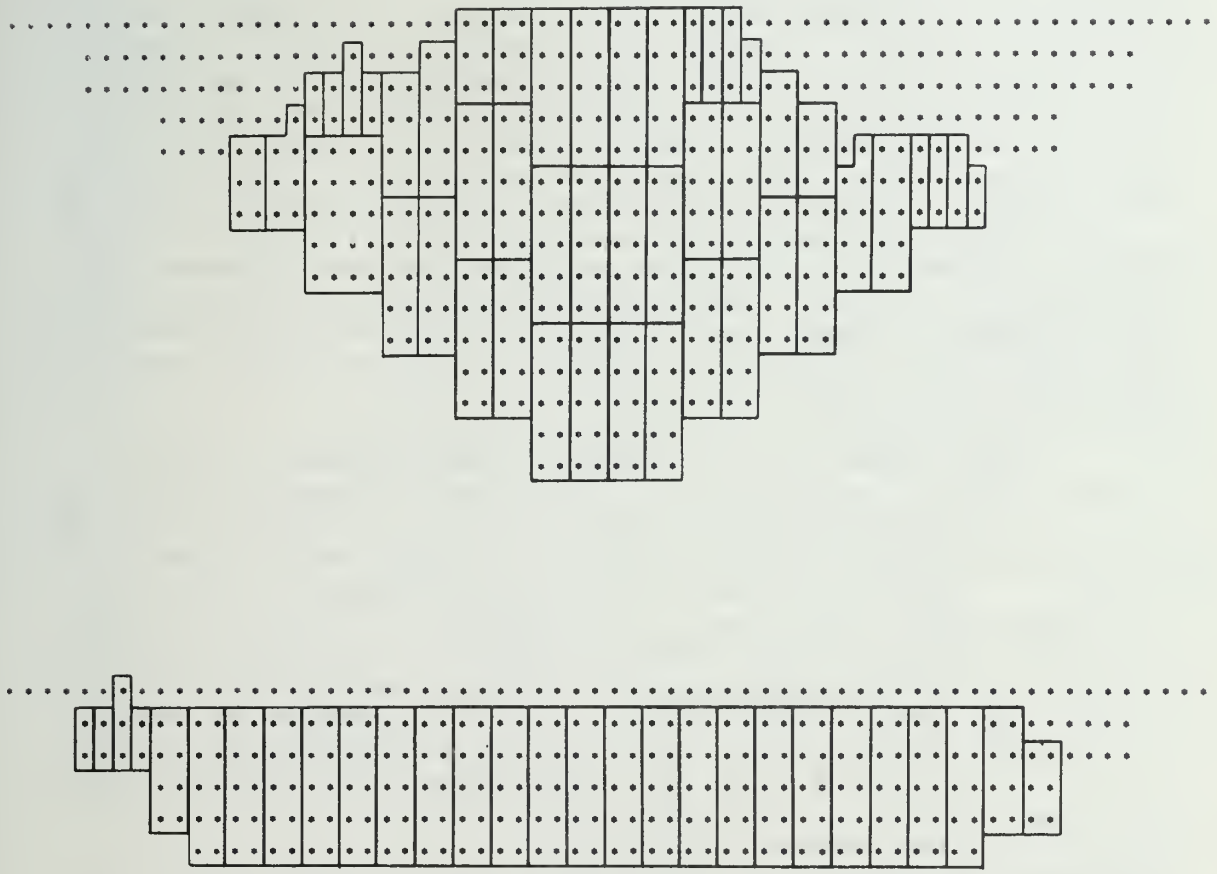


Figure 6.3 32 x 32 Bit Multiplication with Underutilized (5,5,4) Counters Replaced by (3,2) Counters

# 7. COMPARISON OF SEVERAL SCHEMES

The algorithm presented earlier for the mechanical partioning of multipliers is well suited to solution via computer and has, in fact, been programmed to generate partitions for multipliers using various combinations of counters and multiplier modules. The counters and multiplier modules involved are all currently available as off the shelf items in TTL implementation. Hence we may use the characteristics of the individual components in conjunction with the component counts produced by the respective multiplier generating programs to obtain realistic measures of speed, power consumption, relative circuit board area, and cost for a multiplier as a whole.

Four schemes will be compared in this fashion. They are:

1) Generate partial products with 4 x 4 multiply modules and reduce with both (5,5,4) and (3,2) counters, using the (3,2) counters wherever possible (similar to Figure 6.3).
2) Generate matrix with 4 x 4 multiply modules and reduce with both (7,3) and (3,2) counters, using the (3,2) counters wherever possible.
3) Generate matrix with 4 x 4 multiply modules and reduce with (3,2) counters exclusively.
4) Generate matrix with 1 x 1 multiply modules and reduce with (3,2) counters exclusively.

The components involved are in practice implemented in several versions of TTL (i.e., standard TTL, Schottky clamped TTL, and high speed

TTL) hence, for purposes of normalization, their propogation delays are characterized in terms of equivalent logic levels when modeling the multipliers. For purposes of reference, all logic levels may be taken as the standard TTL gate delay of 10 nsec. The 1 x 1 multiplier is characterized as having 2 logic levels--a NAND followed by an inversion--and the (3,2) counter as having 3 logic levels--a level of inversion followed by a 2 level AND-OR-INVERT tree. The 4 x 4 multiplier, as well as the (5,5,4) and (7,3) counters are characterized as ROMs having 6 logic levels--3 levels of row address inversion and decoding and one level each for bit sensing, column selection, and output buffering. An attempt has also been made to normalize the power consumption to that characteristic of a standard TTL implementation of the ICs. It is assumed that the circuits are packaged in standard dual inline packages (DIPs) and that several components are housed in a single DIP when possible (e.g., two (3,2) counters or four 1 x 1 multipliers in a single DIP). This permits us to obtain a figure for the total area occupied by the ICs themselves, which is a good indication of the total area of a circuit board for a given multiplier relative to other multipliers. The cost figures were taken from current distributors catalogs, but are somewhat unrealistic in that IC prices fluctuate with the particular distributor, the quantity involved, the current state of the market, etc. A table of the characteristics of the various ICs is given in Figure 7.1 and the results of the modeling of the various schemes are presented in Figures 7.2 through 7.5.

| | Logic Levels | # Per Pkg. | Pin Count | Package Area | Power in m.w. | Cost |
|---|---|---|---|---|---|---|
| (3,2) | 3 | 2 | 14 | .21 | 220 | 4.00 |
| (7,3) | 6 | 1 | 16 | .24 | 650 | 8.00 |
| (5,5,4) | 6 | 1 | 16 | .24 | 850 | 25.00 |
| 1 x 1 | 2 | 4 | 14 | .21 | 113 | .40 |
| 4 x 4 | 6 | 1 | 20 | .30 | 850 | 12.50 |



Figure 7.1 Characteristics of Some Multipliers and Counters

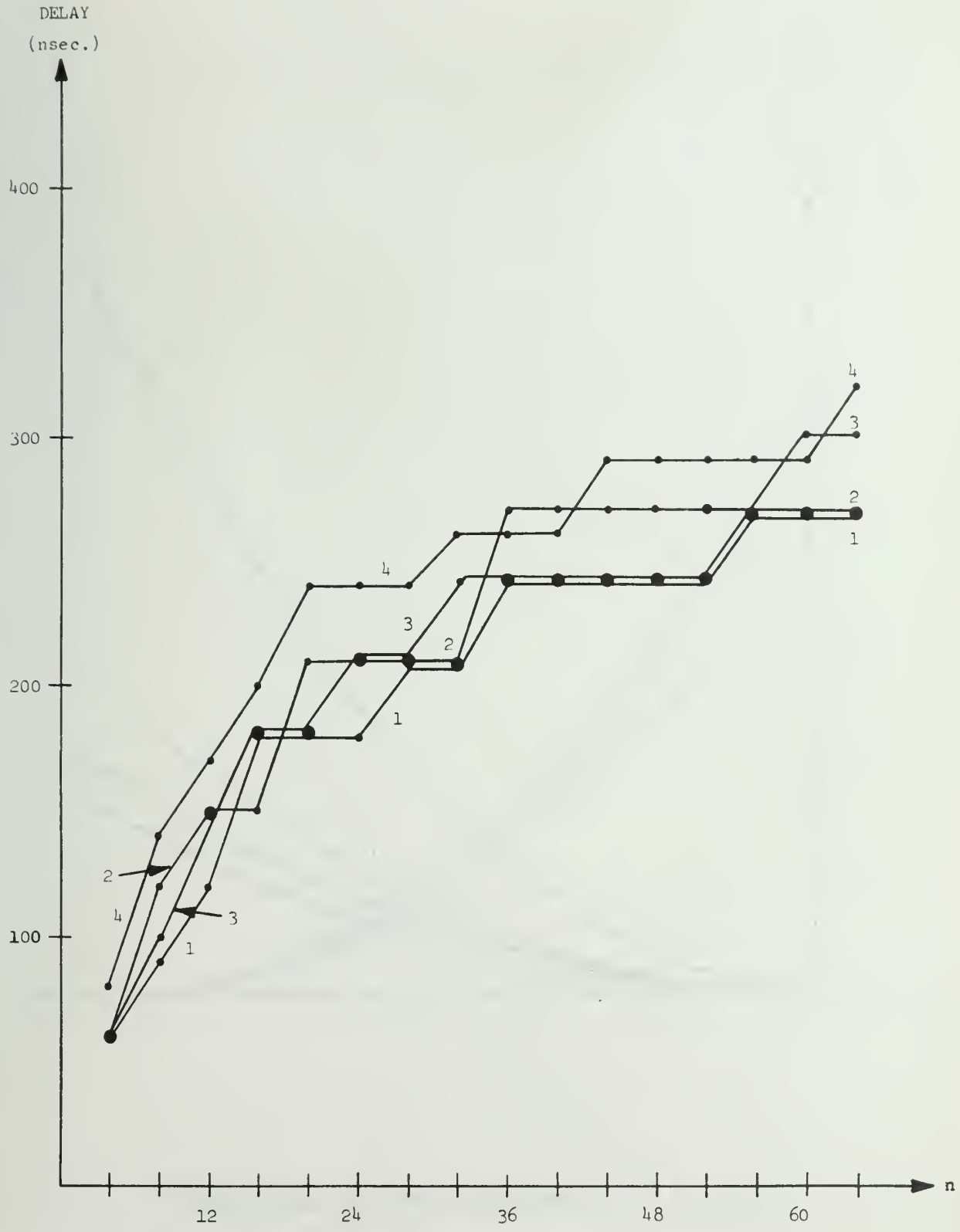



Figure 7.2 Plot of Propagation Delay vs. Word Size

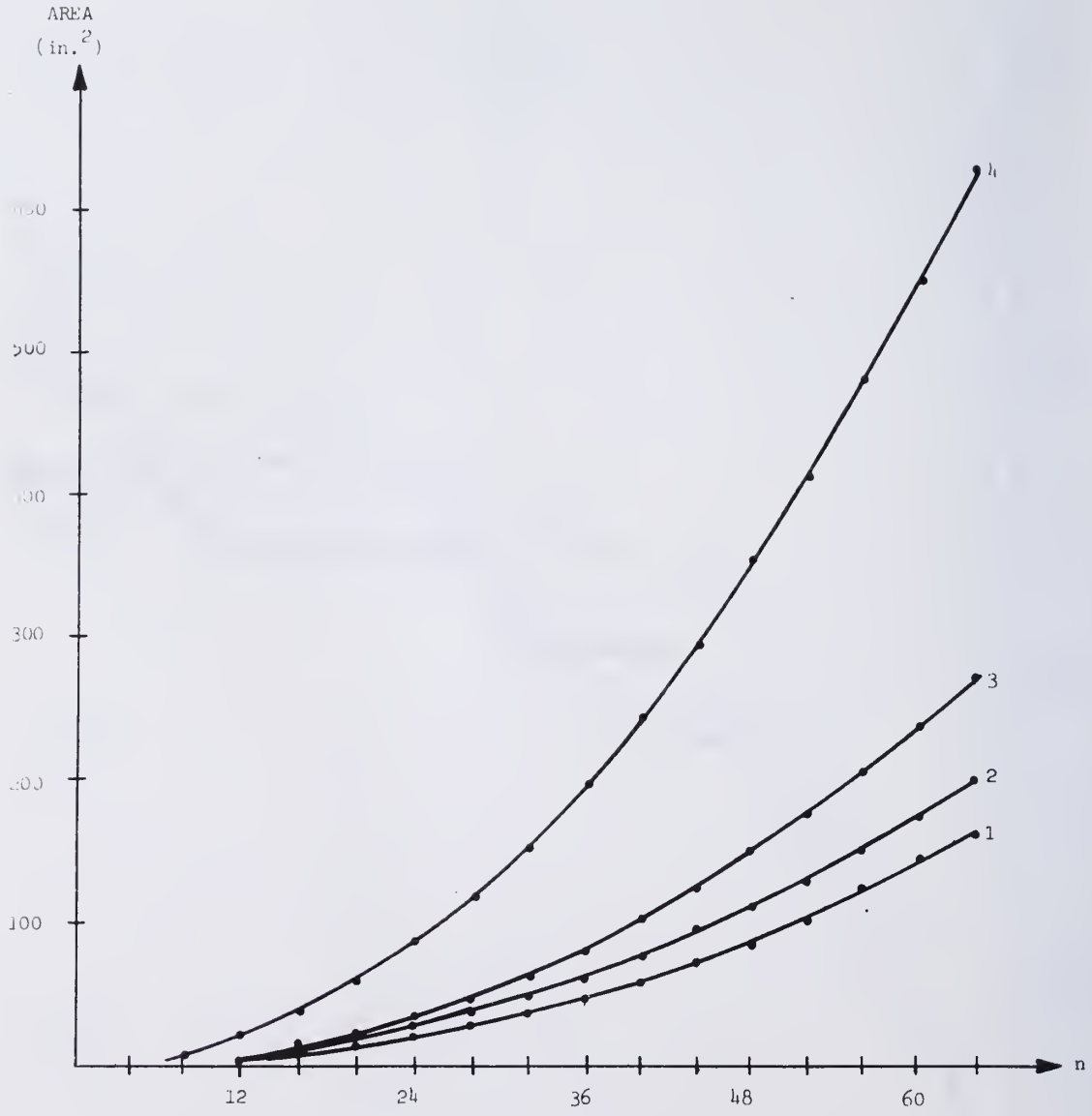



Figure 7.3 Plot of Package Area vs. Word Size

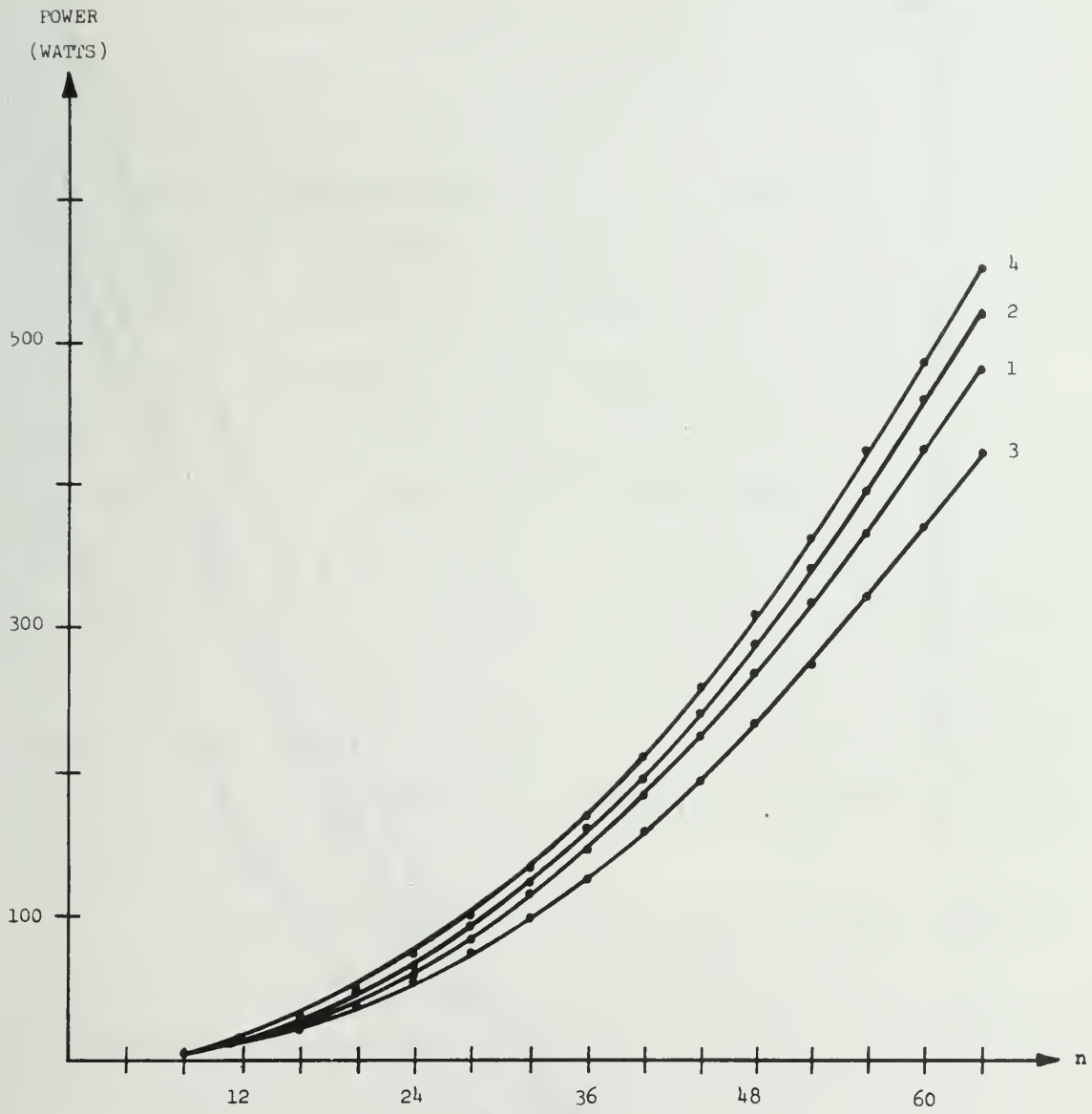



Figure 7.4 Plot of Power Dissipation vs. Word Size

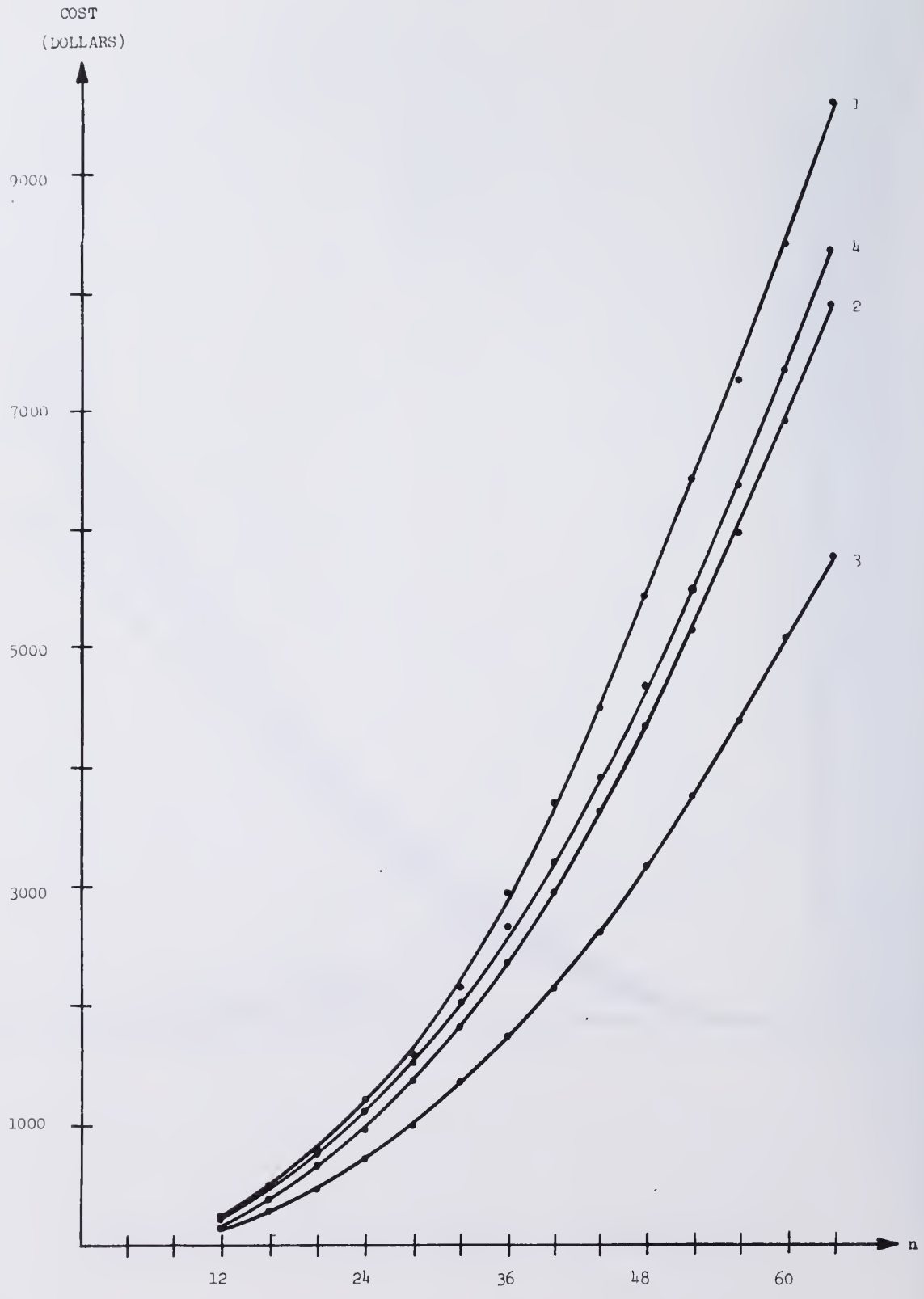



Figure 7.5 Plot of Cost vs. Word Size

## 8. IMPLEMENTATION OF MULTIPLIERS AND COUNTERS

There are several avenues of approach to the implementation of multipliers and counters. The logic for the desired function may be implemented directly by means of gates, or the function may be simulated using read only memory techniques. Direct implementation is feasible in counters only when the height of the input columns are small, (e.g., the traditional full adder (3,2) counter) and carry look ahead adder ((2,2,2, 3,5) counter). Additionally, small units may be used as functional modules in constructing larger units by means of either LSI or hybrid techniques. Figure 8.1 shows a (15,4) counter synthesized via (3,2) and (5,5,4) counters, and a (3,3,3,3,6) counter synthesized via (3,2) and (2,2,2,3,5) counters. This approach may also be applied to large multipliers--an 8 x 8 multiplier may be synthesized from 4 x 4 multipliers and carry look ahead adders using hybrid technology and arrays of gated full adders have been used in fabricating monolithic multipliers as large as 8 x 8 bits. [Breuer]

Table lookup schemes are practical for m x m multipliers and larger counters. In its purest form this type of scheme utilizes a standard ROM with one address bit corresponding to each input bit of the counter or multiplier and every input configuration addressing a distinct word in memory. A ROM simulating a device with i input bits and j output bits must have a total of $2^i$ x j bits of memory. The implementation of multipliers and counters via standard ROMs is especially attractive in that the time and expense of designing and developing a new chip are

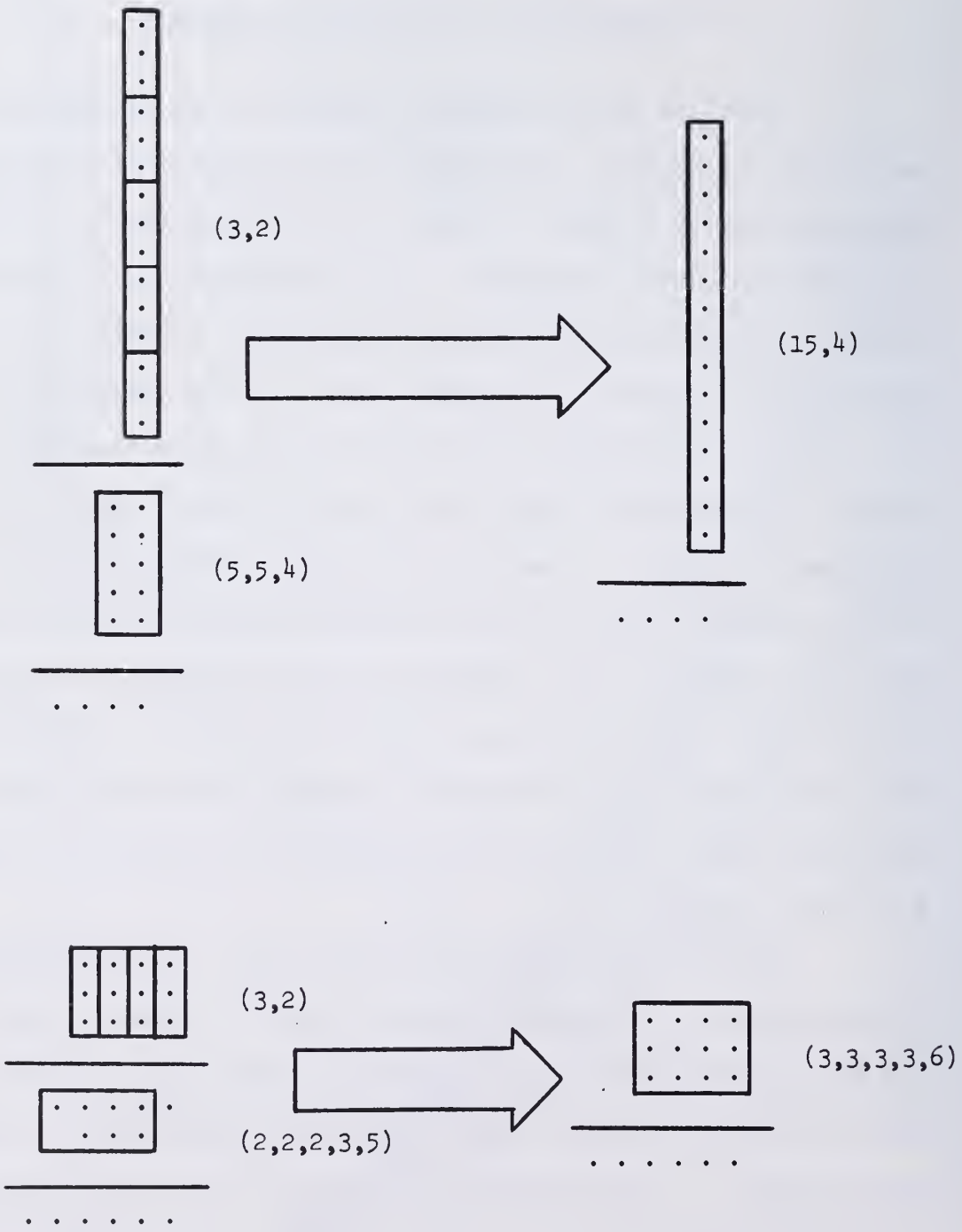



Figure 8.1 Synthesis of (15,4) and (3,3,3,3,6) Counters

avoided--one need only program the appropriate pattern into an existing type of chip. High speed TTL memories are currently available in organizations as large as 1k x 4 and 256 x 8 permitting the implementation of such units as 4 x 4 multipliers, (7,3) counters, (5,5,4) counters, (10,4) counters, and (2,2,3,3,6) counters. The propagation delay of these units is simply the addressing delay of the memory--typically 6 - 8 logic level delays.

One drawback of the standard ROM approach is the high degree of redundancy in the stored information. Consider the case of a ROM implemented (7,3) counter. There are seven distinct configurations of inputs which correspond to the output value 1 meaning that the same output word, 1, must be stored at seven distinct locations. Clearly a reduction in the amount of storage required could be achieved if the input configurations could somehow be mapped into classes such that each member of a class referenced the same memory location. A scheme for such a mapping which incorporates residue threshold functions has been proposed by Ho and Chen. [Ho] The scheme was presented in conjunction with single input column counters (they demonstrate its applicability to the (7,3) counter) but it is also useful for certain multiple column counters as will be shown in the ensuing sketch of their scheme.

The standard ROM implementation of a (5,5,4) counter requires 10 address bits and 4 output bits for a total of

$$2^{10} \times 4 = 4096$$

bits of memory. The internal organization of the memory utilizes some

form of coincidence addressing, e.g., row and column selection. Let the row addresses of the memory correspond to the 32 possible bit configurations of the low order input column of the counter, and the column addresses of the memory correspond to the 32 configurations of the high order input column of the counter. This means that every row address represents a sum of either 0, 1, 2, 3, 4, or 5 in the low order input column and that every column address corresponds to a sum of either 0, 2, 4, 6, 8, or 10 in the high order input column. Note that the OR of all row address lines corresponding to a particular sum conveys as much information about that sum as the constituent address lines, and similarly for the column addresses. Hence, by ORing appropriate address lines we may condense 32 lines into 6 lines, as shown in Figure 8.2, and rather than a 32 x 32 matrix for each output bit we now have a 6 x 6 matrix. The intersection of the row and column lines corresponds to their sum, hence coincident selection of a row line with value $v_r$ and a column line with value $v_c$ implies an output value v of

$$v = v_r + v_c$$

Note that a particular output bit $f_i$ with weighting $2^i$ will contain a 1 for a given value v if and only if

$$2^i \leq (v) \bmod 2^{i+1}$$

e.g., $f_0$ is 1 for 1, 3, 5, 7, ...and $f_1$ is 1 for 2, 3, 6, 7, 12, 13, ... This means that the memory matrix for a given output bit $f_i$ may be programmed by testing each $v_r$ and $v_c$ line for the condition

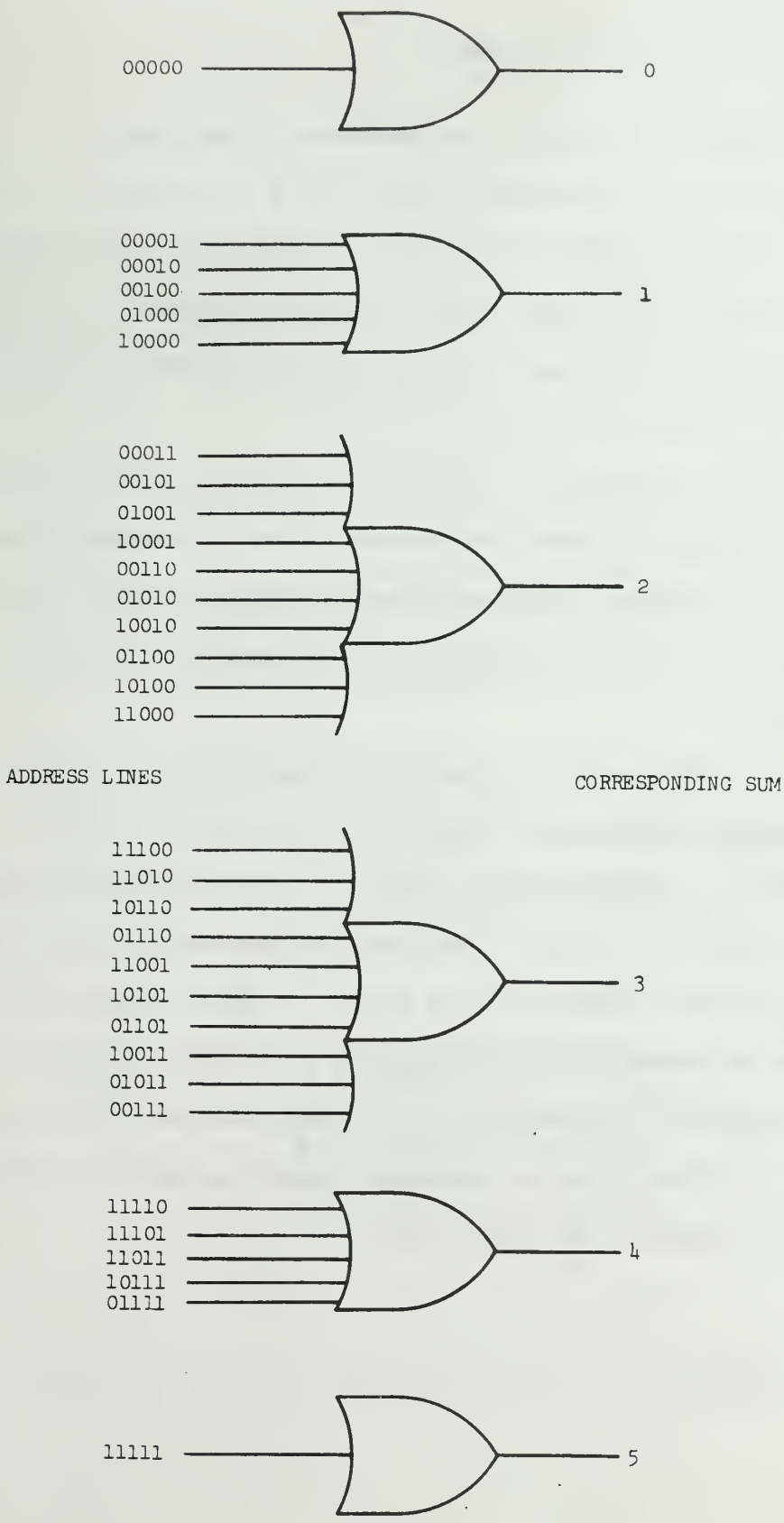



Figure 8.2 Combining Redundant Addresses

$$2^i \leq (v_r + v_c) \bmod 2^{i+1}$$

and inserting a 1 at their intersection if the condition is satisfied and a 0 otherwise. The complete matrix for a (5,5,4) counter programmed in this fashion is shown in Figure 8.3. Since bit $f_0$ is independent of the high order input column its matrix may be replaced by the OR of row lines 1, 3, and 5. The amount of storage required, then, is

$$6 \times 6 \times 3 = 108$$

bits. This represents a savings by a factor of nearly 40 over the 4096 bits required for direct implementation at the expense of one extra logic level in decoding. Some technologies allow this extra logic level to be implemented with no significant extra proagation delay, e.g., ECL collector dotting.

Note that this type of implementation bears great similarity to programmed logis arrays (PLAs). In fact, the most basic form of a PLA consists of a stage in which inputs are selectively ANDed together to form a number of product terms and a second stage in which the products are selectively ORed to form a number of sum of product terms. The condensed implementation discussed above may then be viewed as simply a first PLA which encodes the input column into the proper value, followed by either a ROM or a second PLA which effectively adds the column values to produce the output word.

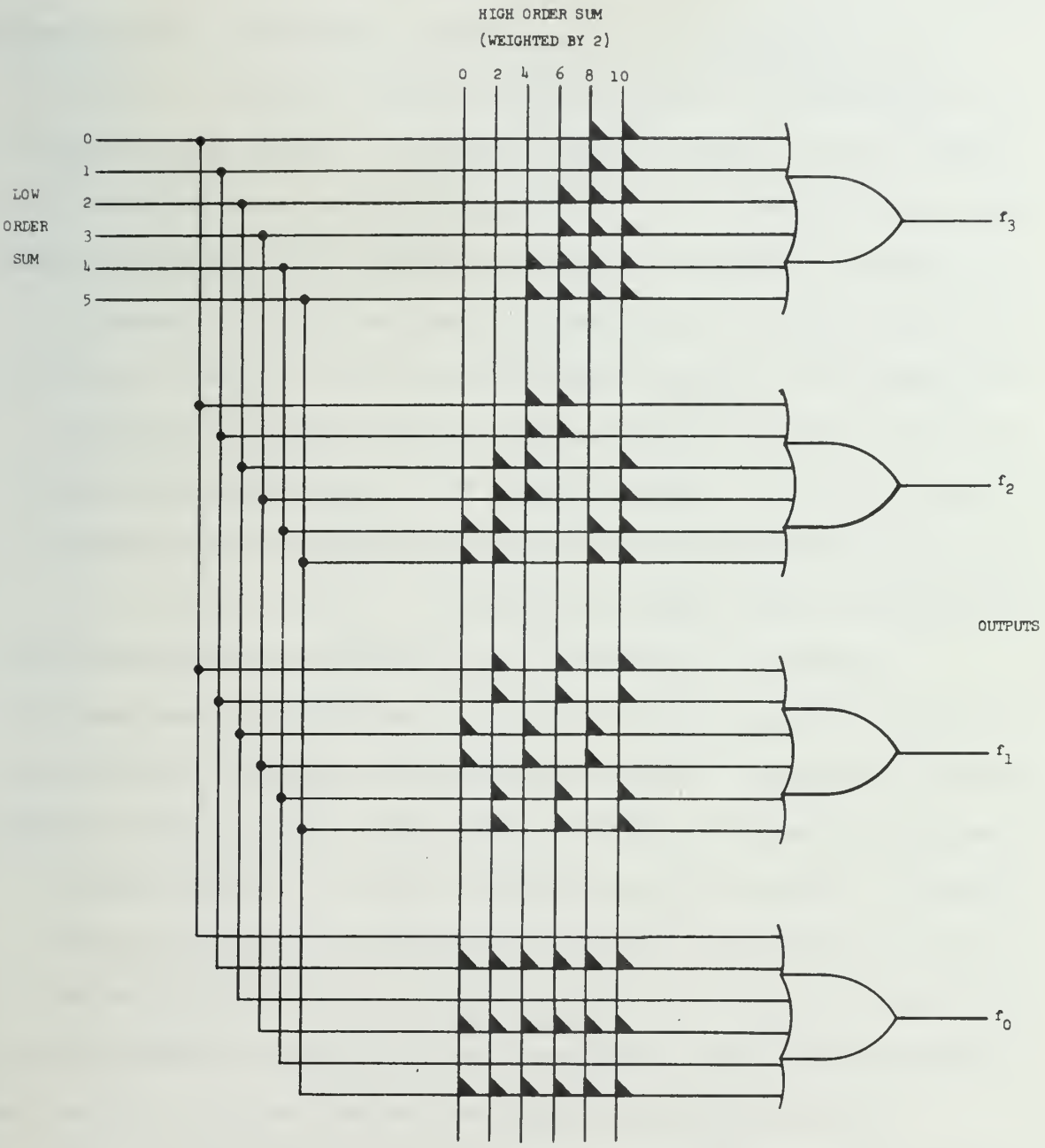



Figure 8.3 PLA-Type Implementation of (5,5,4) Counter

# 9. PROTOTYPE MULTIPLIER

A prototype 24 x 24 bit multiplier incorporating ROM implemented 4 x 4 multipliers and (5,5,4) counters as well as standard (2,2,2,3,5) counters has been fabricated as a portion of a floating point arithmetic unit. The arithmetic unit is a processing element for a large scale array computer considered in a design feasibility study sponsored by NASA. For a description of the processing element and the global aspects of the computer see [Graham]. The multiplier circuit was hand optimized to reduce the number of (5,5,4) counters and to minimize the propagation delay. The design is shown in Figure 9.1. Note that the carries from the (2,2,2,3,5) counters are propagated horizontally in some cases, as indicated by arrows between the counters.

Schottky TTL integrated circuits were used, necessitating care in coping with the high power consumption and fast rise times characteristic of the Schottky family. The high power consumption requires that the $V_{CC}$ and ground distribution system be able to carry large DC currents (about 10A.) and exhibit low inductance in the 10 - 100 MHz range in order to minimize the transients characteristic of saturating logic. The fast rise time of the logic necessitates care in board layout to avoid transmission line effects and crosstalk between reduction levels.

Due to fabrication limitations, we were restricted to two-sided boards with maximum dimensions of 15 x 18 in. The multiplier circuit contains 90 ICs, hence a 15 x 18 in. board allows three square inches of board area per package. While this is normally sufficient area for

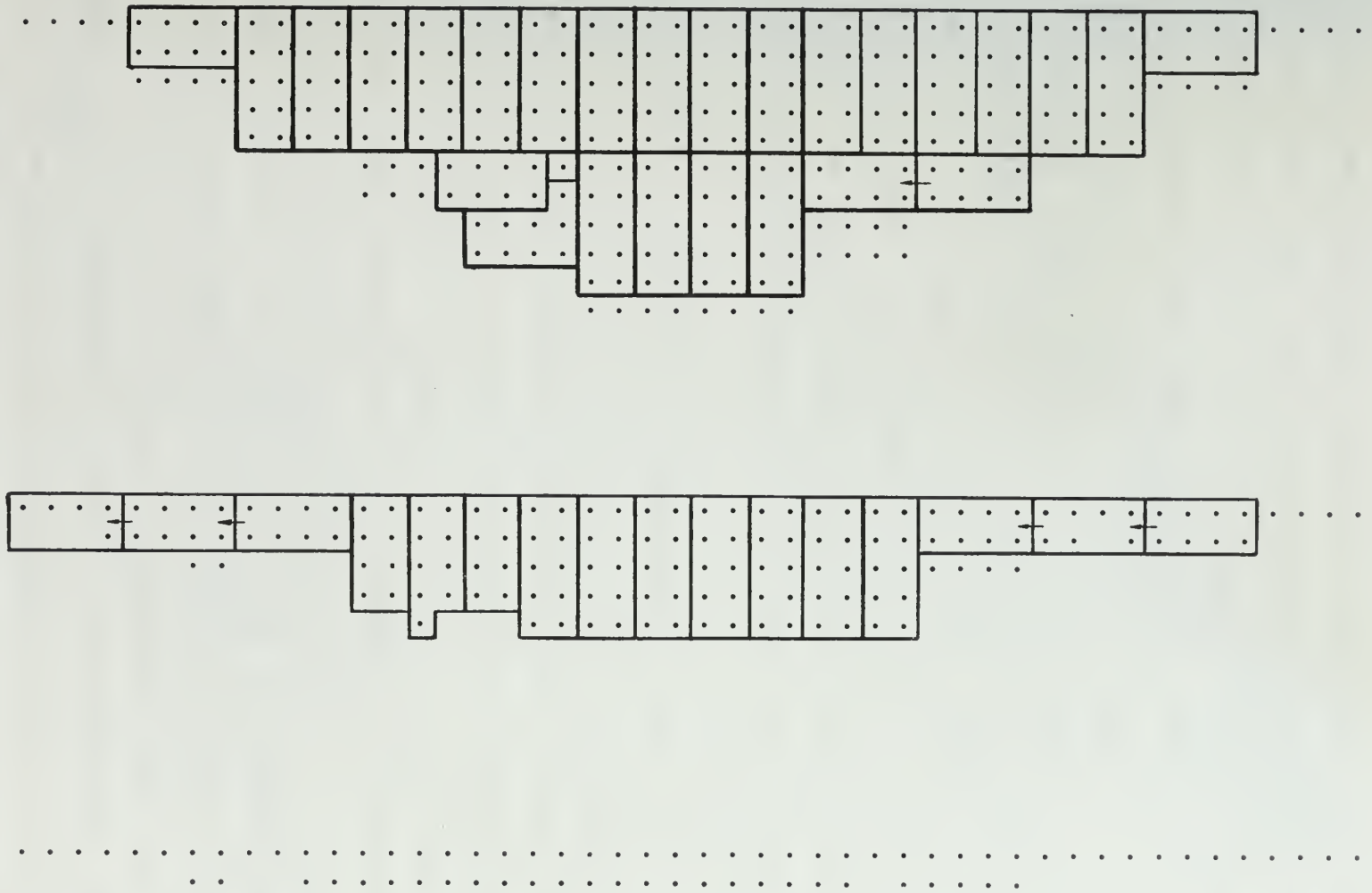
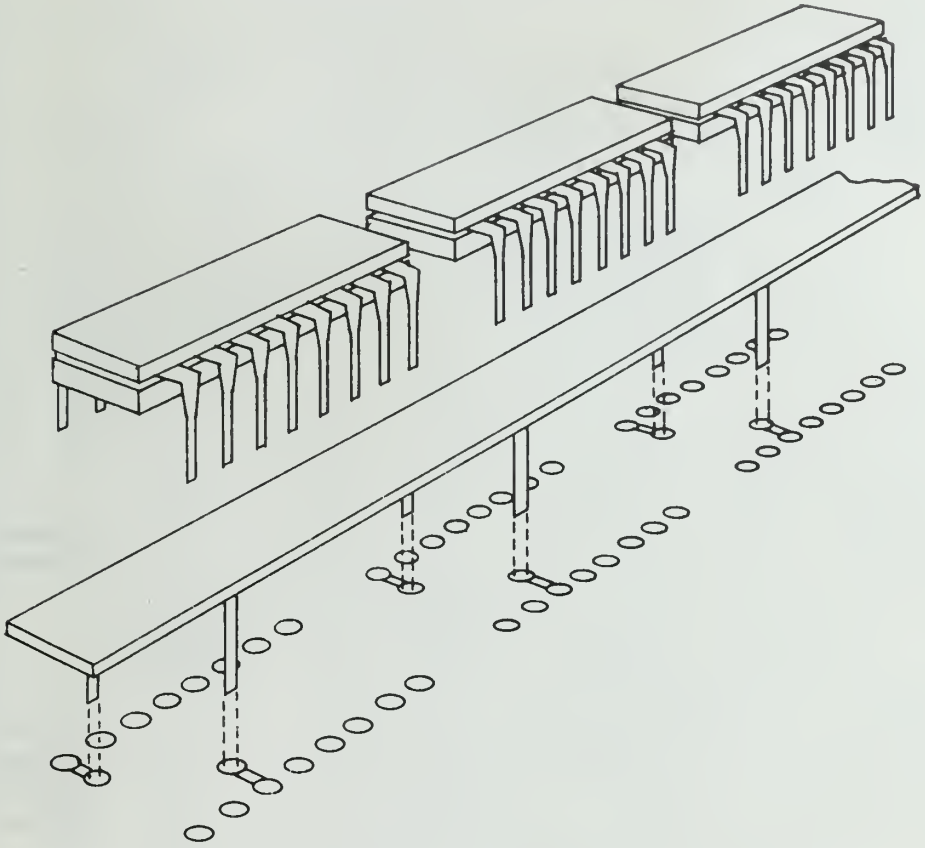




Figure 9.1 24 x 24 Bit Multiplier

standard TTL, it was feared that the power bussing requirements and the complexity of the package interconnections would force a two board layout. In view of this, a novel power distribution system was devised by Mr. Frank Serio; two thin sheets of copper were etched, one for power, one for ground, to form a one piece system of thin strips which lie underneath the rows of integrated circuits and between their pins. This power bussing system need not be etched onto the PC board, allowing higher density and greater flexibility in the board layout. Figure 9.2 shows a schematic and a cross section of the bus system. A trial layout of the nine most tightly interconnected integrated circuits was attempted with promising results, and a decision was made to fabricate the entire circuit on one board.

The layout of the board was done entirely by hand, the package placement being settled upon after several increasingly successful iterations. Due to its complexity, the interconnection specification for the board's artwork was in the form of a wiring list rather than the usual logic diagram form. In its final form, the board consists of a horseshoe type arrangement of integrated circuits with the input lines running up the center of the horseshoe and the output lines running down the outside, as shown in Figures 9.3 and 9.4. The thirty-six 4 x 4 multiplier ROMs immediately surround the input lines and are fed by horizontal connections on the back side of the board, as shown in Figures 9.5 and 9.6. The packages for the first level of reduction surround the 4 x 4 multiplier ROMs and are in turn surrounded by the second level and the carry look ahead adders with their associated carry propagate units. This layout



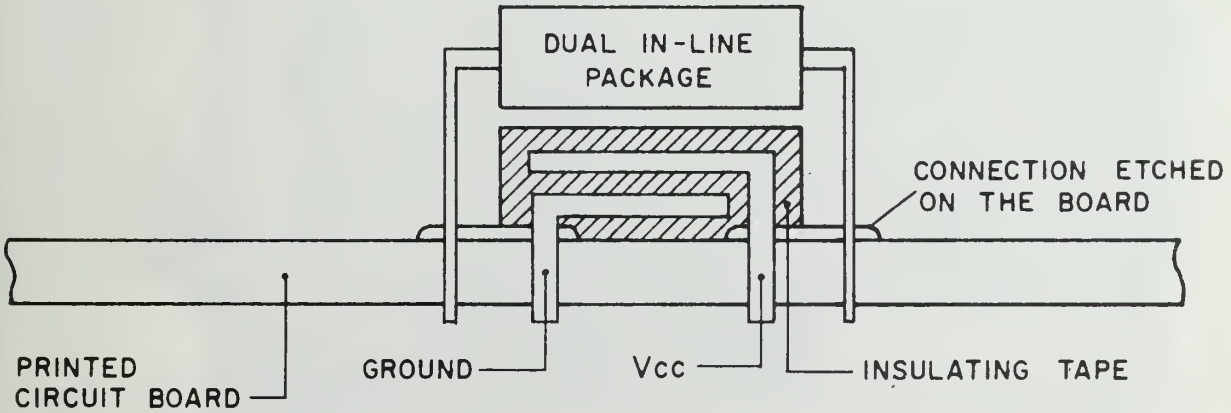



Figure 9.2 Details of Power and Ground Bussing System

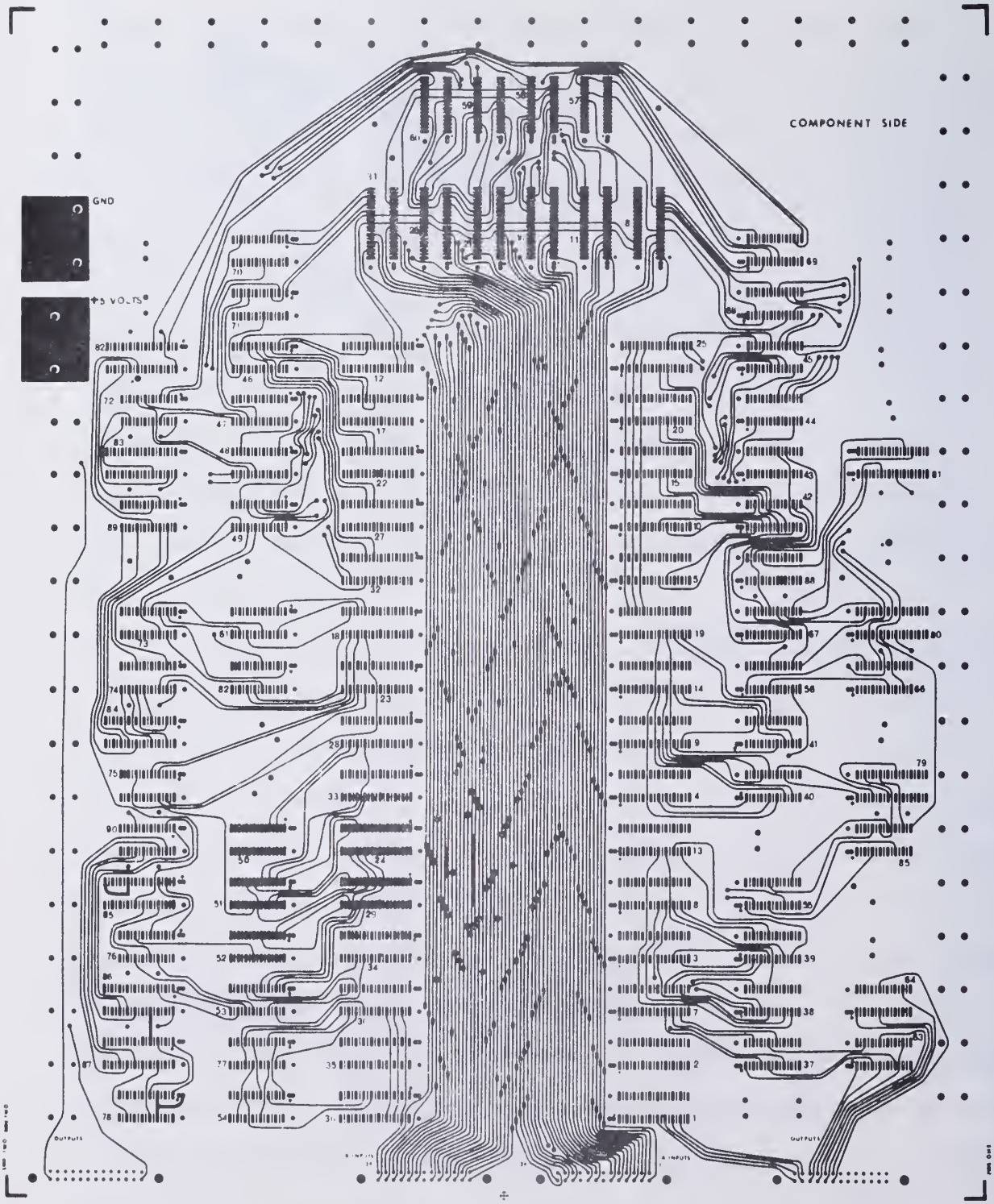



Figure 9.3 Multiplier Prototype Artwork; Component Side

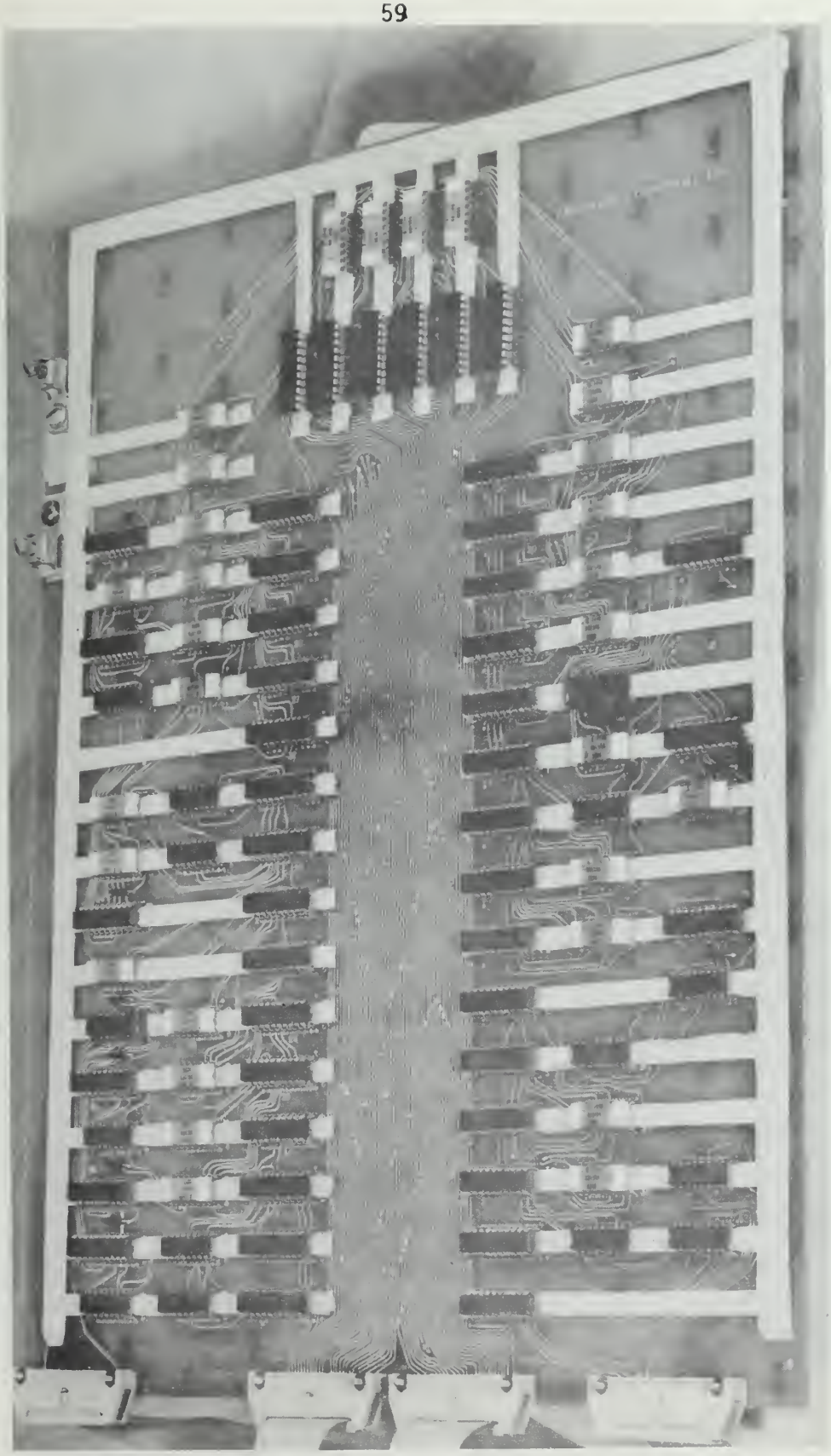


Figure 9.4 Photograph of Mutliplier Prototype; Compondnent Side

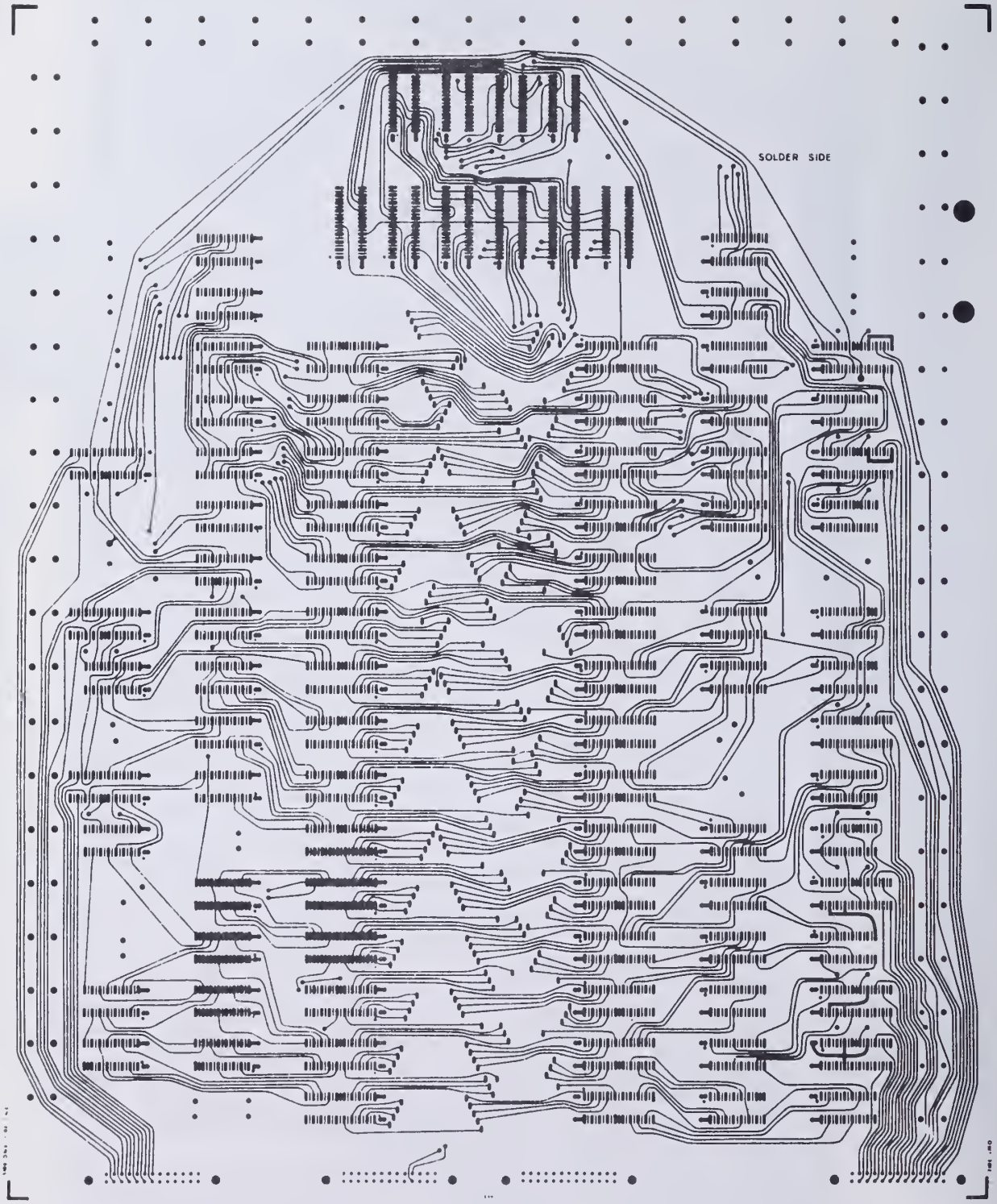



Figure 9.5 Multiplier Prototype Artwork; Solder Side

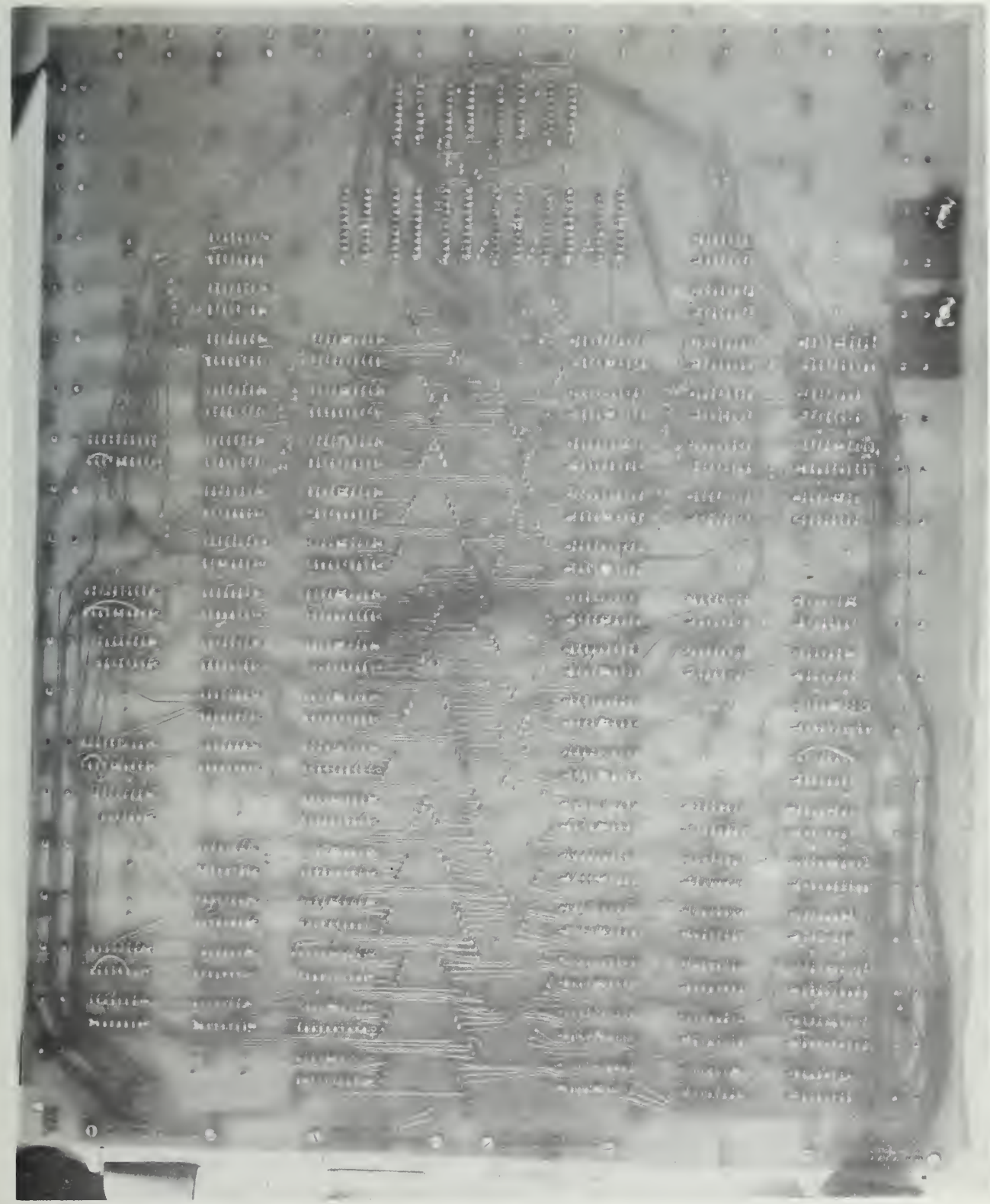


Figure 9.6 Photograph of Multiplier Prototype; Solder Side

tends to minimize signal path length and also limits crosstalk between reduction levels because the signals flow uniformly from the center of the board toward the edges.

# 10. TESTING OF THE PROTOTYPE MULTIPLIER

The testing of the multiplier proceeded in two phases--one phase in which the steady state integrity of the circuit was established and a second in which the dynamic aspects of the circuit were examined.

The only problem uncovered in the first phase of testing was the omission of approximately ten connections to the ground bussing system--this was a minor problem and was quickly remedied. Since there are $2^{48}$ possible input configurations, direct verification of the correctness of the outputs of the circuit by exhaustive testing is not feasible. In fact, at the rate of one microsecond per test on the order of nine years of continuous testing would be required. Instead, a set of inputs for which the outputs are easily calculated by hand were used. For example, multiplications in which one of the operands contains only one nonzero bit or multiplications of two contiguous blocks of '1' bits. This constitutes a good test of the correctness of the signal interconnections on the circuit board, but does not provide exhaustive testing of the contents of the read only memories. Hence the ROMs were screened in advance by means of a master-slave type test and several defective ROMs were discarded.

The dynamic phase of the testing investigated such parameters as the propagation delay of the circuit, the power and ground noise and the rise times and transient behavior of the signals. An interface unit was constructed to facilitate manual input of the two operands and visual inspection of the results via toggle switches and LEDs. Both the

inputs and outputs were buffered via D-type registers. This permits determination of the propagation delay of the circuit by clearing the input register, clocking the data into the input register, and, after an adjustable interval, clocking the output register. The interval between clocking the input and output registers is adjusted to the smallest value such that the correct answer is displayed on the LEDs. A block diagram and photograph of the test interface unit appears in Figures 10.1 and 10.2, respectively. The clear and clock signals are generated by employing a simple ring counter to split a master clock into three out-of-phase clocks, each at one-third the frequency of the master. The relationship of the control signals is shown in Figure 10.3.

It should be noted that this scheme includes input and output register settling times as well as delays in the cabling between the multiplier board and the buffer registers when measuring the propagation delay. Initial measurements of the worst case delay were about 380 nsec. Inspection of the signal quality, however, revealed very slow rise times and a great deal of crosstalk in the lines between the input register and the multiplier. It was determined that this was due to capacitive loading between signal lines both in the input cables and in the paths feeding the 4 x 4 multiplier ROM stages (i.e., the lines at the center of the horseshoe). The cables employed were approximately 18 in. long flat flexible cables with conductors spaced on .05 in. centers. Alternating grounds, unfortunately, were not employed. The corresponding input lines in the center of the board were also spaced on .05 in. centers, the intervals between paths being about equal to the path width, and

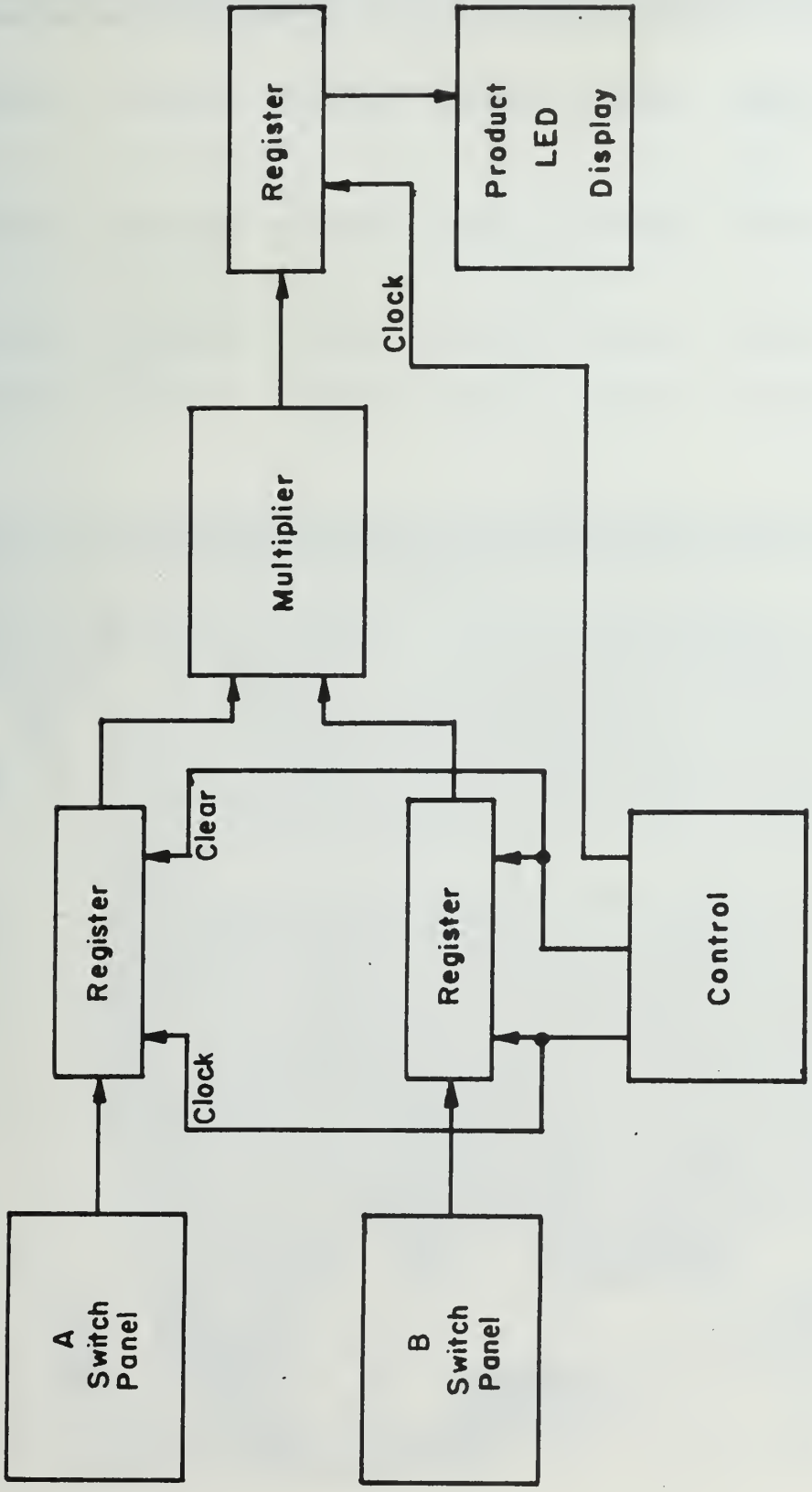



Figure 10.1 Multiplier Prototype Interface Unit
Block Diagram

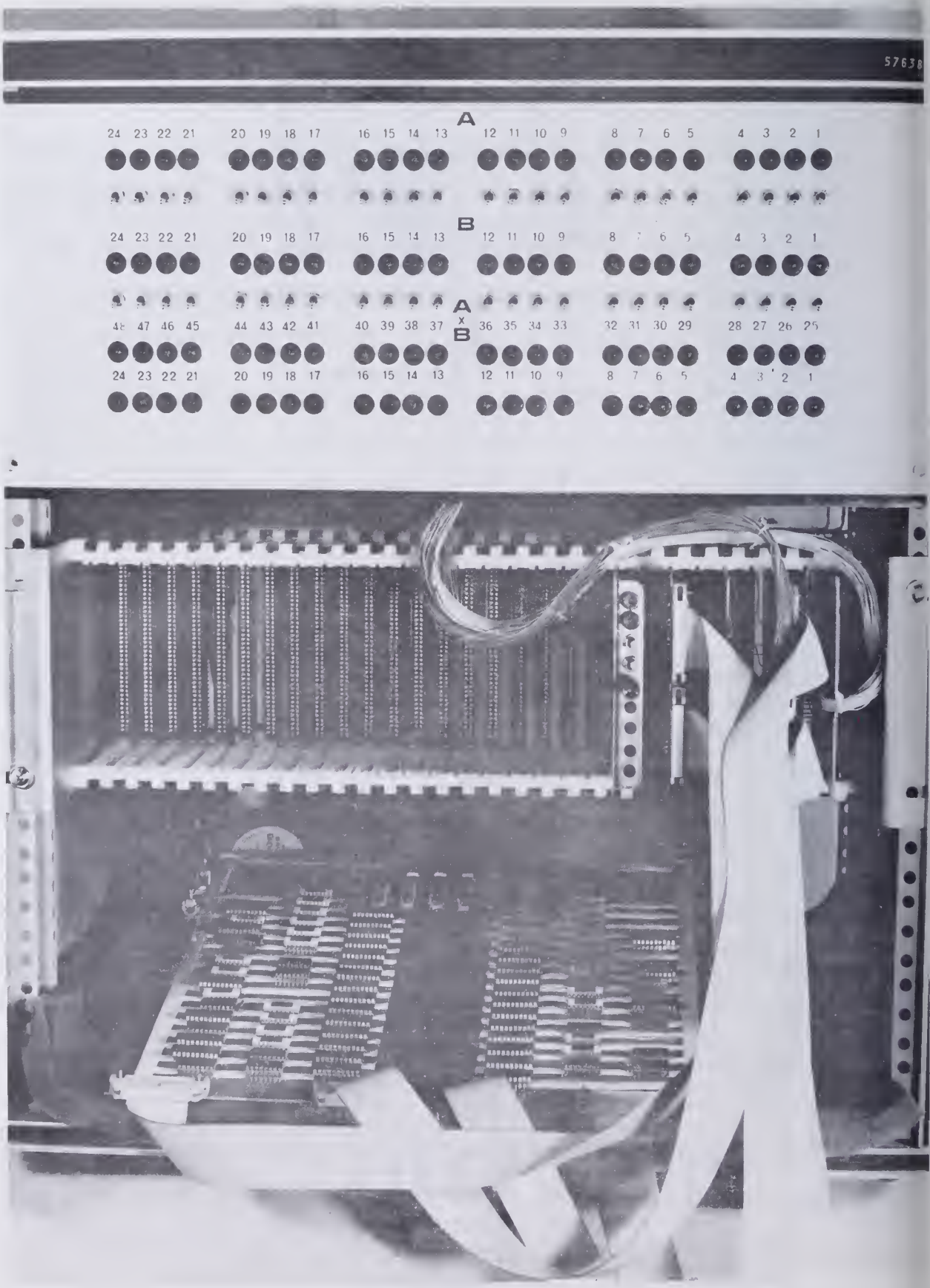



Figure 10.2 Photograph of Multiplier and Interface Unit

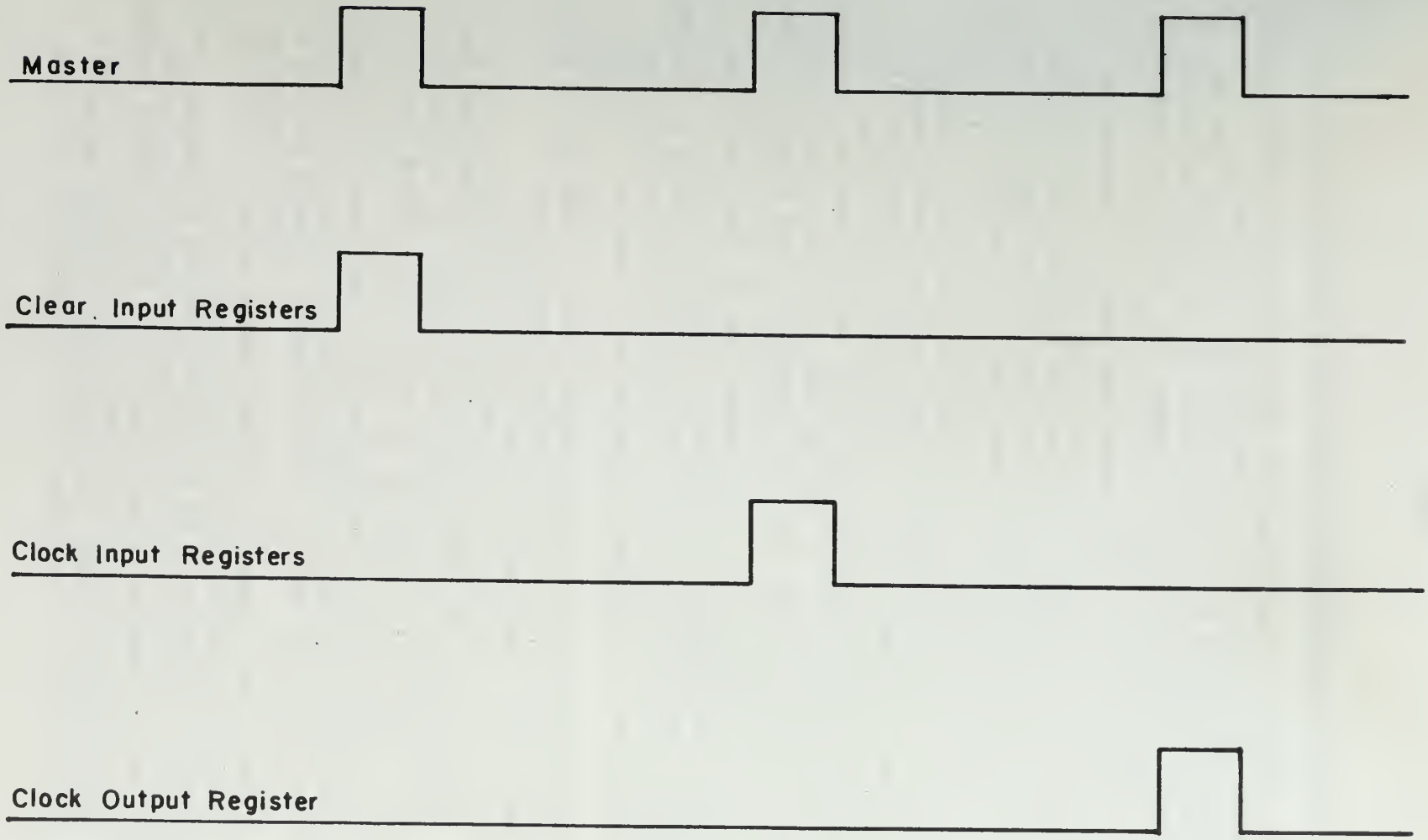



Figure 10.3 Interface Unit Control Signals

again alternating grounds were not used. The crosstalk problem was attacked by placing a ground plane over the input lines on the top side of the board. The ground plane consisted of an insulating layer of electrical tape covered by a layer of copper foil, which was adequately grounded at several points. This reduced the circuit's propagation delay somewhat and improved the risetimes and crosstalk on the inputs so a similar ground plane was placed over the corresponding area on the back side of the board and the input cables were shortened to about 10 in. and shielded with copper foil. Adequate ground returns were also established between the multiplier and buffer circuits and the standard TTL buffer registers were replaced with their high speed Schottky equivalents. The net effect of these improvements was to lower the propagation delay to about 200 nsec. The sum of the typical delay as specified by the manufacturer for components along the worst case path is about 190 nsec, while the sum of the rated maximum delays is 260 nsec. This figure is well within the expected range of performance.

Inspection of the power distribution system during operation revealed very little noise, attesting to the efficacy of the bussing system. The signal quality on intra-board paths was found to be excellent with fast rise times and very little ringing, although the 4 x 4 multiplier ROMs tended to generate spurious output pulses as their addresses changed. This seems to be an inherent feature of the memories and is not a serious problem although the circuit would probably operate slightly faster (by 10 - 20 nsec.) had the pulses not been present.

# 11. CONCLUDING REMARKS

The partial product matrix generation-reduction schemes of Wallace and Dadda may be enhanced through the use of multiplier modules larger than 1 x 1 bit and counters larger and more general than (3,2) counters. The larger multiplier modules permit the generation of a partial product matrix containing fewer total bits and having a maximum height less than that generated by 1 x 1 multipliers and, with current implementations, require fewer IC packages to do so. Larger, more complex counters permit reduction of matrices in fewer stages of counters with fewer IC packages than the simpler (3,2) counters while maintaining a similar number of total logic levels. The net result is a multiplier with the speed of Dadda's scheme and the compactness of current implementation of array multipliers. [Advanced Micro Devices 25S05, Fairchild 9344]

Several large counters and multiplier modules have been realized as TTL integrated circuits and are also feasible as ECL circuits. The feasibility of their use in multiplication schemes has been demonstrated by the fabrication of a 200 ns. 24 x 24 bit prototype. Advances in LSI and hybrid technology should make possible still larger counters and multiplier modules.

The algorithm for the logical design of multipliers incorporating generalized counters and multiplier modules is extremely straightforward. The physical implementation and maintenance of such multipliers should be facilitated by their compactness and hence should require less

time and expense than comparable multipliers employing Dadda's scheme, especially for large (e.g., 64 bits) words. This type of multiplier then, is attractive for a variety of applications, ranging from fast floating point mini and midi computers to large scientific machines.

# LIST OF REFERENCES

Advanced Micro Devices, Data Book, 1974.

Baugh, C. R. and Wooley, B. A., "Two's Complement Parallel Array Multiplication Algorithm," IEEE Trans. Comput., Vol C-22, pp 1045-1047, Dec. 1973.

Breuer, D. R., "A High Speed Monolithic 8 x 8 Multiplier," unpublished correspondence.

Chung, T. J. and Bedrosian, S. D., "Fast Digital Multiplier Based on Iterative Cellular Arrays of ROMs," unpublished correspondence.

Dadda, L., "Some Schemes for Parallel Multipliers," Alta Frequenza, Vol 19, pp 349-356, May 1965.

Deegan, I. D., "Cellular Multiplier for Signed Binary Numbers," Electronics Letters, Vol 7, pp 436-437, July 29, 1971.

Fairchild Semiconductor, TTL Data Book, June 1972.

Graham, M. L., "The Design and Logic Simulation of an Array Computer," Ph.D. Thesis, University of Illinois, 1975.

Habibi, A. and Wintz, P. A., "Fast Multipliers," IEEE Trans. Comput., Vol C-19, pp 153-157, Feb. 1970.

Ho, I. T. and Chen, T. C., "Multiple Addition by Residue Threshold Functions and Their Representation by Array Logic," IEEE Trans. Comput., Vol C-22, pp 762-767, Aug. 1973.

Kingsbury, N. G., "High Speed Binary Multiplier," Electronics Letters, Vol 7, pp 277-278, May 20, 1971.

Majithia, J. C. and Kitai, R., "An Iterative Array for Multiplication of Signed Binary Numbers," IEEE Trans. Comput., Vol C-20, pp 214-216, Feb. 1971.

Pezaris, S. D., "A 40 ns 17-Bit by 17-Bit Array Multiplier," IEEE Trans. Comput., Vol C-20, pp 442-447, April 1971.

Singh, S. and Waxman, R., "Multiple Operand Addition and Multiplication," IEEE Trans. Comput., Vol C-22, pp 113-120, Feb. 1973.

Svoboda, A., "Adder With Distributed Control," IEEE Trans. Comput., Vol C-19, pp 749-751, Aug. 1970.

Texas Instruments, Supplement to the TTL Data Book, 1974.

Wallace, C. S., "A Suggestion for a Fast Multiplier," IEEE Trans. Elect. Comput., Vol EC-13, pp 14-17, Feb. 1964.

| BIBLIOGRAPHIC DATA SHEET | 1. Report No. UIUCDCS-R-75-756 | 2. | 3. Recipient's Accession No. |
|---|---|---|---|

4. Title and Subtitle

A CLASS OF COMPACT HIGH SPEED PARALLEL MULTIPLICATION SCHEMES

5. Report Date: September 1975

6.


7. Author(s): William John Stenzel

8. Performing Organization Rept. No.

9. Performing Organization Name and Address
Department of Computer Science
University of Illinois
Urbana, Illinios 61801


10. Project/Task/Work Unit No.


11. Contract/Grant No.: US NASA NAS5-23334


12. Sponsoring Organization Name and Address
National Aeronautics and Space Administration
Goddard Space Flight Center
2880 Broadway
New York, New York 10025

13. Type of Report & Period Covered: Master of Science Thesis

14.

15 Supplementary Notes

16. Abstracts


Parallel multiplication schemes are divisible into 2 classes - iterative cellular array schemes and Dadda-type schemes in which a partial product matrix is generated and then reduced by means of pseudo adders. Array schemes are more compact than traditional forms of generation-reduction schemes but are much slower for large operands. Current LSI technology has made possible ICs which can form smaller partial product matrices as well as ICs which can reduce larger portions of matrices than traditional pseudo adders. This makes possible generation-reduction type multipliers which rival array multipliers in compactness. This paper deals with such compact forms of generation-reduction multipliers. An algorithm for their design is presented and a prototype 24 x 24 bit multiplier is presented.


17. Key Words and Document Analysis. 17a. Descriptors


Parallel multipliers
Dadda's multiplier
Wallace tree reduction
Read only memories
Multiple operand addition


17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

| 18. Availability Statement | 19. Security Class (This Report) | 21. No. of Pages |
|---|---|---|
| Release Unlimited | UNCLASSIFIED | 79 |
| | 20. Security Class (This Page) UNCLASSIFIED | 22. Price -------- |

**INSTRUCTIONS FOR COMPLETING FORM NTIS-35 (10-70)** (Bibliographic Data Sheet based on COSATI Guidelines to Format Standards for Scientific and Technical Reports Prepared by or for the Federal Government, PB-180 600).

1. **Report Number.** Each report shall carry a unique alphanumeric designation. Select one of the following types: (a) alphanumeric designation provided by the sponsoring agency, e.g., **FAA-RD-68-09;** or, if none has been assigned, (b) alphanumeric designation established by the performing organization e.g., **FASEB-NS-87;** or, if none has been established, (c) alphanumeric designation derived from contract or grant number, e.g., **PH-43-64-932-4.**

2. Leave blank.

3. **Recipient's Accession Number.** Reserved for use by each report recipient.

4. **Title and Subtitle.** Title should indicate clearly and briefly the subject coverage of the report, and be displayed prominently. Set subtitle, if used, in smaller type or otherwise subordinate it to main title. When a report is prepared in more than one volume, repeat the primary title, add volume number and include subtitle for the specific volume.

5. **Report Date.** Each report shall carry a date indicating at least month and year. Indicate the basis on which it was selected (e.g., date of issue, date of approval, date or preparation.

6. **Performing Organization Code.** Leave blank.

7. **Author(s).** Give name(s) in conventional order (e.g., John R. Doe, or J.Robert Doe). List author's affiliation if it differs from the performing organization.

8. **Performing Organization Report Number.** Insert if performing organization wishes to assign this number.

9. **Performing Organization Name and Address.** Give name, street, city, state, and zip code. List no more than two levels of an organizational hierarchy. Display the name of the organization exactly as it should appear in Government indexes such as **USGRDR-I.**

10. **Project/Task/Work Unit Number.** Use the project, task and work unit numbers under which the report was prepared.

11. **Contract/Grant Number.** Insert contract or grant number under which report was prepared.

12. **Sponsoring Agency Name and Address.** Include zip code.

13. **Type of Report and Period Covered.** Indicate interim, final, etc., and, if applicable, dates covered.

14. **Sponsoring Agency Code.** Leave blank.

15. **Supplementary Notes.** Enter information not included elsewhere but useful, such as: Prepared in cooperation with . . . Translation of . . . Presented at conference of . . . To be published in . . . Supersedes . . . Supplements . . .

16. **Abstract.** Include a brief (200 words or less) factual summary of the most significant information contained in the report. If the report contains a significant bibliography or literature survey, mention it here.

17. **Key Words and Document Analysis. (a). Descriptors.** Select from the Thesaurus of Engineering and Scientific Terms the proper authorized terms that identify the major concept of the research and are sufficiently specific and precise to be used as index entries for cataloging.
**(b). Identifiers and Open-Ended Terms.** Use identifiers for project names, code names, equipment designators, etc. Use open-ended terms written in descriptor form for those subjects for which no descriptor exists.
**(c). COSATI Field/Group.** Field and Group assignments are to be taken from the 1965 COSATI Subject Category List. Since the majority of documents are multidisciplinary in nature, the primary Field/Group assignment(s) will be the specific discipline, area of human endeavor, or type of physical object. The application(s) will be cross-referenced with secondary Field/Group assignments that will follow the primary posting(s).

18. **Distribution Statement.** Denote releasability to the public or limitation for reasons other than security for example "Release unlimited". Cite any availability to the public, with address and price.

19 & 20. **Security Classification.** Do not submit classified reports to the National Technical Information Service.

21. **Number of Pages.** Insert the total number of pages, including this one and unnumbered pages, but excluding distribution list, if any.

22. **Price.** Insert the price set by the National Technical Information Service or the Government Printing Office, if known.